ANDES Manual

Release 2.0.0

Hantao Cui

Mar 12, 2026

CONTENTS

1 About ANDES **1**
- 1.1 Symbolic-Numeric Framework . . . 1
- 1.2 What's Included . . . 1
- 1.3 Citation . . . 2
- 1.4 Acknowledgments . . . 2
- 1.5 License . . . 2

2 Tutorials **3**
- 2.1 Learning Paths . . . 3
  - 2.1.1 New User . . . 3
  - 2.1.2 Power System Analyst . . . 3
  - 2.1.3 Model Developer . . . 4
- 2.2 Installation . . . 4
  - 2.2.1 Quick Install . . . 4
  - 2.2.2 New to Python . . . 4
    - Setting Up miniforge . . . 4
  - 2.2.3 Using uv . . . 5
  - 2.2.4 Extra Packages . . . 5
  - 2.2.5 Development Install . . . 6
  - 2.2.6 Updating ANDES . . . 6
  - 2.2.7 Troubleshooting . . . 7
    - Multiple Copies Installed . . . 7
    - Windows DLL Error . . . 7
  - 2.2.8 Next Steps . . . 7
- 2.3 First Simulation . . . 7
  - 2.3.1 Import ANDES . . . 8
  - 2.3.2 Load a Test Case . . . 8
  - 2.3.3 Run Power Flow . . . 9
  - 2.3.4 Run Time-Domain Simulation . . . 9
  - 2.3.5 Plot Results . . . 10
  - 2.3.6 Run Eigenvalue Analysis . . . 11
  - 2.3.7 Key Concepts . . . 13
  - 2.3.8 Using the Command Line . . . 14
  - 2.3.9 Cleanup . . . 14
  - 2.3.10 Next Steps . . . 14

2.4 Power Flow Analysis . . . 15
2.4.1 Setup . . . 15
2.4.2 Running Power Flow . . . 15
Command Line Alternative . . . 16
2.4.3 Accessing Results . . . 16
Bus Results . . . 16
Generator Results . . . 18
Line Flows . . . 18
Exploring Bus Connections . . . 18
2.4.4 Configuration . . . 19
2.4.5 Troubleshooting . . . 20
Non-Convergence . . . 20
Viewing Iteration Progress . . . 20
2.4.6 Power Flow Report . . . 22
2.4.7 Example: Formatted Bus Voltage Display . . . 23
2.4.8 Cleanup . . . 23
2.4.9 Next Steps . . . 23
2.5 Time-Domain Simulation . . . 24
2.5.1 Setup . . . 24
2.5.2 Basic Simulation Workflow . . . 24
Command Line Alternative . . . 25
2.5.3 Adding Disturbances . . . 25
Three-Phase Faults . . . 26
Line and Device Trips (Toggle) . . . 28
Parameter Changes (Alter) . . . 30
Altering Dynamic vs. Static Models . . . 32
2.5.4 Configuration Options . . . 33
2.5.5 Accessing Results . . . 34
Using `get_timeseries()` . . . 34
Advanced: Direct DAE Access . . . 35
2.5.6 Plotting Results . . . 35
Exporting Results . . . 37
2.5.7 Multi-Stage Simulation . . . 37
2.5.8 Output Files . . . 39
2.5.9 Troubleshooting . . . 40
Simulation Fails to Converge . . . 40
Checking Initialization . . . 40
Flat Run Test . . . 42
2.5.10 Cleanup . . . 42
2.5.11 Next Steps . . . 42
2.6 Data and File Formats . . . 42
2.6.1 Setup . . . 43
2.6.2 Loading ANDES Excel Files . . . 43
2.6.3 Loading PSS/E Files . . . 44
Command Line Usage . . . 45
Cross-Format Loading . . . 45
Converting to ANDES Format . . . 45
2.6.4 Inspecting Parameters . . . 45

Viewing Model Parameters . . . 45
Getting Specific Parameters . . . 46
Accessing Parameter Arrays . . . 46
2.6.5 Modifying Parameters . . . 47
Using set() with device base . . . 47
Direct Array Modification . . . 48
Verifying Modified Parameters . . . 48
2.6.6 Modifying Device Status . . . 48
2.6.7 Saving Modified Cases . . . 49
2.6.8 Accessing Variables After Simulation . . . 50
Power Flow Results . . . 50
Variable Types . . . 50
2.6.9 Working with Device Indices . . . 51
Listing Device Indices . . . 51
Converting Index to Internal UID . . . 51
2.6.10 Parameter Units . . . 52
2.6.11 Cleanup . . . 52
2.6.12 Next Steps . . . 52
2.7 Plotting Results . . . 52
2.7.1 Setup and Run Simulation . . . 53
2.7.2 The Plotter Object . . . 54
2.7.3 Plotting by Variable . . . 54
Plotting a Subset of Variables . . . 55
2.7.4 Index-Based Plotting . . . 57
2.7.5 Customizing Plots . . . 58
Labels and Legends . . . 58
Scaling Values . . . 59
Grid and Greyscale . . . 60
Adding Curves to an Existing Figure . . . 61
2.7.6 Interactive Plots with Plotly . . . 62
2.7.7 Saving Figures . . . 63
2.7.8 Exporting to CSV . . . 64
2.7.9 Plot Method Reference . . . 65
2.7.10 Cleanup . . . 66
2.7.11 Next Steps . . . 66
2.8 Eigenvalue Analysis . . . 66
2.8.1 Setup . . . 66
2.8.2 Running Eigenvalue Analysis . . . 67
2.8.3 Plotting Eigenvalues on the S-Plane . . . 69
2.8.4 Eigenvalue Report . . . 70
2.8.5 Parameter Sweeps and Root Loci . . . 71
2.8.6 Cleanup . . . 72
2.8.7 Next Steps . . . 73
2.9 Parameter Sweeps and Batch Processing . . . 73
2.9.1 Setup . . . 73
2.9.2 File-Based Batch Processing . . . 73
Generating Case Files . . . 74
Running Cases in Parallel . . . 74

Returning System Objects for Post-Processing . . . 77
2.9.3 In-Memory Parameter Sweeps . . . 80
Power Flow Parameter Sweep . . . 80
Batch Time-Domain Simulation . . . 82
2.9.4 Choosing the Right Approach . . . 85
2.9.5 Cleanup . . . 86
2.9.6 Next Steps . . . 86
2.10 Contingency Analysis . . . 86
2.10.1 Setup . . . 87
2.10.2 N-1 Line Contingency Screening . . . 87
2.10.3 Stability Assessment . . . 88
2.10.4 Visualizing Contingency Results . . . 89
2.10.5 Fault-and-Clear Analysis . . . 92
2.10.6 Critical Clearing Time . . . 94
2.10.7 Stability Metrics Summary . . . 95
2.10.8 Cleanup . . . 95
2.10.9 Next Steps . . . 96
2.11 Dynamic Control and Setpoint Changes . . . 96
2.11.1 Setup . . . 96
2.11.2 Initialize the System . . . 96
2.11.3 Understanding Turbine Governor Setpoints . . . 97
2.11.4 Stage 1: Run Simulation to Steady State . . . 98
Finding a Generator by Bus Number . . . 98
2.11.5 Stage 2: Apply Setpoint Change . . . 99
2.11.6 Stage 3: Restore Setpoint . . . 102
2.11.7 Applications . . . 104
2.11.8 Cleanup . . . 105
2.11.9 Next Steps . . . 105
2.12 Frequency Response and Load Shedding . . . 105
2.12.1 Setup . . . 106
2.12.2 Simulating a Generator Trip . . . 106
2.12.3 Determining the Generation Shortfall . . . 109
2.12.4 Implementing Load Shedding . . . 109
2.12.5 Key Concepts . . . 112
2.12.6 Cleanup . . . 113
2.12.7 Next Steps . . . 113
2.13 State Estimation . . . 113
2.13.1 Setup . . . 114
2.13.2 Running State Estimation . . . 115
2.13.3 Custom Measurements . . . 116
Selective Meter Placement . . . 117
Low-Level `add()` Method . . . 118
2.13.4 Zero-Noise Verification . . . 118
2.13.5 Bad Data Detection . . . 119
2.13.6 Robust Estimation with LAV . . . 121
2.13.7 Multi-Island Systems . . . 123
2.13.8 Configuration . . . 124
2.13.9 Cleanup . . . 125

2.13.10 Next Steps . . . 125
2.14 Continuation Power Flow . . . 125
2.14.1 Setup . . . 126
2.14.2 Running CPF . . . 127
2.14.3 Plotting the Nose Curve . . . 128
2.14.4 Parameterization Methods . . . 131
2.14.5 Termination Control . . . 133
2.14.6 Full Curve Tracing (Upper and Lower Branches) . . . 134
2.14.7 Per-Device Loading Targets . . . 136
2.14.8 QV Curve Analysis . . . 138
2.14.9 Configuration . . . 140
2.14.10 Reactive Power Limits . . . 140
2.14.11 Cleanup . . . 144
2.14.12 Next Steps . . . 144
2.15 Inspecting Model Equations . . . 144
2.15.1 Setup . . . 144
2.15.2 Listing Available Models . . . 144
2.15.3 Preparing Model Equations . . . 145
2.15.4 Model Documentation . . . 146
2.15.5 Accessing Variables . . . 149
2.15.6 Symbolic Equations . . . 150
Differential Equations . . . 150
Algebraic Equations . . . 151
Jacobian Matrices . . . 151
2.15.7 Services . . . 152
2.15.8 Practical Applications . . . 153
2.15.9 See Also . . . 153
2.16 Reinforcement Learning with ANDES . . . 153
2.16.1 Setup . . . 154
2.16.2 Constructing the Environment . . . 154
Observation Specification . . . 155
Action Specification . . . 156
Reward Function . . . 156
2.16.3 Running an Episode . . . 157
2.16.4 Adding Disturbances . . . 158
2.16.5 Comparing Control Policies . . . 159
2.16.6 Deterministic Reset . . . 159
2.16.7 Training with Stable-Baselines3 . . . 160
2.16.8 Accessing the System Object . . . 161
2.16.9 TDS Configuration . . . 161
2.16.10 Summary . . . 162
2.16.11 Next Steps . . . 162

3 Gallery **163**
3.1 Running the Examples . . . 166
3.1.1 Critical Clearing Time Mapping . . . 167
System Setup . . . 167
CCT Sweep . . . 168

Stability Boundary . . . 169
Spatial Distribution of CCT . . . 171
Rotor Angle Trajectories at the Stability Boundary . . . 173
Extensions . . . 177
3.1.2 Low-Inertia Frequency Stability . . . 177
System Setup . . . 178
Frequency Response Under Varying Inertia . . . 178
COI Frequency After Generator Trip . . . 179
Quantitative Metrics . . . 180
RoCoF vs. Inertia Level . . . 180
Extend This Example . . . 181
3.1.3 Forced Oscillation Source Localization . . . 182
System Setup . . . 182
Periodic Disturbance Injection . . . 183
Observing the Forced Oscillation . . . 184
Spectral Analysis . . . 185
Energy-Based Source Localization . . . 186
Phase-Based Confirmation . . . 187
Extend This Example . . . 188
3.1.4 Physics-Informed Surrogate from Symbolic Equations . . . 188
Step 1 — Extract Symbolic Equations . . . 189
Step 2 — Eliminate Algebraic Variables . . . 190
Step 3 — Convert to Differentiable PyTorch Functions . . . 191
Step 4 — Train a Parameterized PINN . . . 192
Results . . . 198
Extend This Example . . . 201
3.1.5 Post-Fault Rotor Oscillation on an SMIB System . . . 201
Build the SMIB System . . . 201
Power Flow . . . 203
Apply a Three-Phase Fault and Run TDS . . . 203
Rotor Speed Response . . . 203
Quantify the Oscillation Decay . . . 205
Phase Portrait . . . 206
Terminal Voltage and Exciter Response . . . 207
Toward Supplementary Damping Control . . . 208
RL Environment Setup . . . 209
No-Control Baseline . . . 210
Train a PPO Agent . . . 210
Evaluate the Trained Agent . . . 211
Results . . . 212
Extend This Example . . . 213

4 Modeling **215**
4.1 Who This Is For . . . 215
4.2 Modeling Concepts . . . 215
4.2.1 Key Ideas . . . 215
Hybrid Symbolic-Numeric Framework . . . 215
Atomic Types . . . 216

DAE Formulation . . . 216
System Architecture . . . 216
4.2.2 Hybrid Symbolic-Numeric Framework . . . 216
Key Concept . . . 216
Benefits . . . 216
Framework Components . . . 217
Code Generation . . . 217
Example: Model Definition . . . 218
Symbolic Variables . . . 218
Integration with Solver . . . 219
See Also . . . 219
4.2.3 Atomic Types . . . 219
Value Provider (v-provider) . . . 219
Equation Provider (e-provider) . . . 220
Summary . . . 221
See Also . . . 222
4.2.4 System Architecture . . . 222
System Overview . . . 222
Dynamic Imports . . . 222
Code Generation . . . 223
DAE Storage . . . 223
Decentralized Architecture . . . 226
Jacobian Matrices . . . 227
Calling Model Methods . . . 228
Model Calling Protocol . . . 229
External Variable Protocol . . . 229
Configuration . . . 230
Device Lifecycle . . . 230
Timed Event Mechanism . . . 232
See Also . . . 233
4.2.5 DAE Formulation . . . 233
Mathematical Form . . . 233
Variable Types . . . 234
Power Flow vs Dynamic . . . 234
Jacobian Matrices . . . 235
Equation Conventions . . . 235
Initialization . . . 236
Handling Discontinuities . . . 236
Mass Matrix . . . 237
Eigenvalue Analysis . . . 237
See Also . . . 237
4.3 Model Components . . . 237
4.3.1 Parameters . . . 238
Background . . . 238
Parameter Types . . . 238
NumParam . . . 249
IdxParam . . . 252
ExtParam . . . 254

DataParam . . . 255
TimerParam . . . 256
BaseParam . . . 258
Parameters in Equations . . . 260
Mandatory vs Optional . . . 260
Input Validation . . . 260
See Also . . . 261
4.3.2 Variables . . . 261
Background . . . 261
Variable Types . . . 262
State . . . 274
Algeb . . . 275
External Variables . . . 276
Value and Equation Strings . . . 277
Flags for Value Overwriting . . . 279
Observable . . . 280
Alias Variables . . . 281
BaseVar . . . 282
Common Patterns . . . 283
See Also . . . 284
4.3.3 Services . . . 284
Background . . . 284
Service Types . . . 285
Internal Constants . . . 311
External Constants . . . 314
Shape Manipulators . . . 315
Value Manipulation . . . 319
Idx and References . . . 320
Events . . . 321
Flags . . . 322
Data Select . . . 323
Miscellaneous . . . 324
Common Patterns . . . 325
See Also . . . 326
4.3.4 Discrete Components . . . 326
Background . . . 326
Discrete Types . . . 327
Limiters . . . 351
Comparers . . . 354
Deadband . . . 356
Others . . . 358
Using Flags in Equations . . . 360
Naming Convention . . . 361
Common Patterns . . . 361
See Also . . . 362
4.3.5 Blocks . . . 362
Background . . . 362
Block Types . . . 365

Linear Blocks . . . 399
First Order Blocks . . . 401
Second Order Blocks . . . 408
PI Controllers . . . 410
Saturation . . . 417
Usage Examples . . . 418
Naming Convention . . . 419
See Also . . . 419
4.3.6 Groups . . . 420
Background . . . 420
GroupBase . . . 420
Standard Groups . . . 428
Defining Groups . . . 428
Model Group Registration . . . 429
Using Groups for References . . . 429
BackRef for Connections . . . 429
Group Substitution . . . 430
Common Group Patterns . . . 430
Group Requirements . . . 431
Creating New Groups . . . 431
See Also . . . 431
4.4 Creating Models . . . 431
4.4.1 Examples Overview . . . 431
4.4.2 Workflow . . . 432
4.4.3 Model Structure . . . 432
Basic Structure . . . 432
ModelData Class . . . 433
Model Class . . . 433
Model Flags . . . 434
Component Order . . . 434
Connecting to Other Models . . . 435
Equation Definition . . . 435
Complete Example . . . 436
Registration . . . 437
Code Generation . . . 437
See Also . . . 437
4.4.4 Example: Static Model (Shunt) . . . 437
Background . . . 437
The Shunt Model . . . 438
Step-by-Step Breakdown . . . 439
Key Concepts Illustrated . . . 442
Testing the Model . . . 443
See Also . . . 443
4.4.5 Example: Dynamic Model (BusFreq) . . . 443
Background . . . 443
The BusFreq Model . . . 444
Step-by-Step Breakdown . . . 445
Signal Flow . . . 448

Key Concepts Illustrated . . . 448
Testing the Model . . . 449
Key Differences from Static Model . . . 449
See Also . . . 450
4.4.6 Example: TGOV1 Turbine Governor . . . 450
Model Overview . . . 450
Block Diagram . . . 450
Mathematical Equations . . . 451
Equation-Based Implementation . . . 451
Block-Based Implementation . . . 453
Comparison of Approaches . . . 454
Guidelines . . . 454
Complete Source Code . . . 454
See Also . . . 454
4.4.7 Example: IEEEST Power System Stabilizer . . . 454
Model Overview . . . 455
Block Diagram . . . 455
Data Classes . . . 455
PSSBase Model Class . . . 456
IEEESTModel . . . 458
Final Assembly . . . 460
Model Registration . . . 460
Key Techniques Summary . . . 461
Complete Source Code . . . 461
See Also . . . 461
4.4.8 Testing Models . . . 462
Testing Levels . . . 462
Quick Validation . . . 462
Power Flow Test . . . 463
Time-Domain Test . . . 463
Initialization Test . . . 464
Writing pytest Tests . . . 464
Verification Against Reference . . . 465
Debugging Tips . . . 466
Common Issues . . . 467
See Also . . . 467

5 Reference **469**
5.1 Command Line Interface . . . 469
5.1.1 Basic Usage . . . 469
5.1.2 Commands Overview . . . 469
5.1.3 andes run . . . 470
Basic Examples . . . 470
Routine Options . . . 470
Simulation Control . . . 470
File Options . . . 471
Parallel Execution . . . 471
Interactive Shell . . . 471

Complete Options . . . 472
5.1.4 andes plot . . . 472
Basic Usage . . . 472
Finding Variable Indices . . . 472
Plot Options . . . 473
5.1.5 andes doc . . . 473
5.1.6 andes prepare . . . 473
5.1.7 andes selftest . . . 474
5.1.8 andes misc . . . 474
5.1.9 Verbosity Levels . . . 474
5.1.10 Environment Variables . . . 475
5.1.11 Common Workflows . . . 475
Standard Analysis . . . 475
Batch Studies . . . 475
Convert and Simulate . . . 475
Add Disturbance Sheets . . . 476
5.1.12 Getting Help . . . 476
5.1.13 See Also . . . 476
5.2 Configuration . . . 476
5.2.1 Configuration Resolution . . . 476
5.2.2 Configuration Levels . . . 477
5.2.3 Viewing Configuration . . . 477
All Configuration . . . 477
Specific Options . . . 478
5.2.4 Modifying Configuration . . . 478
In Python . . . 478
At Load Time . . . 478
Command Line . . . 479
5.2.5 Configuration File . . . 479
Config File Format . . . 479
5.2.6 System Options . . . 480
5.2.7 Runtime Options . . . 480
5.2.8 Power Flow Options (PFlow) . . . 480
5.2.9 Time Domain Options (TDS) . . . 481
5.2.10 Model Limit Options . . . 481
5.2.11 Sparse Solvers . . . 481
5.2.12 Numba JIT Compilation . . . 481
5.2.13 Limit Adjustment . . . 482
5.2.14 Logging . . . 482
5.2.15 See Also . . . 482
5.3 Model Reference . . . 483
5.3.1 ACLine . . . 484
Line . . . 484
5.3.2 ACNode . . . 487
Bus . . . 487
5.3.3 ACShort . . . 488
Jumper . . . 489
5.3.4 Calculation . . . 490

ACE . . . 491
ACEc . . . 493
COI . . . 494
5.3.5 Collection . . . 496
Area . . . 496
5.3.6 DCLink . . . 497
Ground . . . 497
R . . . 498
L . . . 499
C . . . 501
RCp . . . 502
RCs . . . 504
RLs . . . 506
RLCs . . . 507
RLCp . . . 509
5.3.7 DCTopology . . . 510
Node . . . 511
5.3.8 DG . . . 512
PVD1 . . . 512
ESD1 . . . 522
EV1 . . . 531
EV2 . . . 540
5.3.9 DGProtection . . . 549
DGPRCT1 . . . 549
DGPRCTExt . . . 555
5.3.10 DataSeries . . . 562
TimeSeries . . . 562
5.3.11 DynLoad . . . 564
ZIP . . . 564
FLoad . . . 566
5.3.12 DynShaft . . . 567
SHAFT5 . . . 568
5.3.13 Exciter . . . 572
EXDC2 . . . 572
IEEEX1 . . . 576
ESDC1A . . . 581
ESDC2A . . . 587
EXST1 . . . 592
ESST3A . . . 595
SEXS . . . 602
IEEET1 . . . 605
EXAC1 . . . 609
EXAC2 . . . 616
EXAC4 . . . 623
ESST4B . . . 627
AC8B . . . 633
IEEET3 . . . 640
ESAC1A . . . 646

ESST1A . . . 653
ESAC5A . . . 660
5.3.14 Experimental . . . 665
5.3.15 FreqMeasurement . . . 666
BusFreq . . . 666
BusROCOF . . . 668
FreqDiv . . . 670
5.3.16 Information . . . 672
Summary . . . 672
5.3.17 Interface . . . 672
Fortescue . . . 673
5.3.18 Motor . . . 676
Motor3 . . . 676
Motor5 . . . 679
5.3.19 OutputSelect . . . 683
Output . . . 683
5.3.20 PLL . . . 683
PLL1 . . . 684
PLL2 . . . 686
5.3.21 PSS . . . 687
IEEEST . . . 688
ST2CUT . . . 693
5.3.22 PhasorMeasurement . . . 699
PMU . . . 699
5.3.23 RenAerodynamics . . . 700
WTARA1 . . . 701
WTARV1 . . . 702
5.3.24 RenExciter . . . 703
REECA1 . . . 704
REECA1E . . . 712
REECA1G . . . 720
REECB1 . . . 728
5.3.25 RenGen . . . 736
REGCA1 . . . 736
REGCP1 . . . 741
REGCV1 . . . 747
REGCV2 . . . 753
REGF1 . . . 757
REGF2 . . . 763
REGF3 . . . 769
5.3.26 RenGovernor . . . 776
WTDTA1 . . . 777
WTDS . . . 780
5.3.27 RenPitch . . . 782
WTPTA1 . . . 782
5.3.28 RenPlant . . . 785
REPCA1 . . . 786
5.3.29 RenTorque . . . 792

WTTQA1 . . . 792
5.3.30 StaticACDC . . . 797
VSCShunt . . . 797
5.3.31 StaticGen . . . 801
PV . . . 801
Slack . . . 804
5.3.32 StaticLoad . . . 806
PQ . . . 806
5.3.33 StaticShunt . . . 809
Shunt . . . 809
ShuntTD . . . 810
ShuntSw . . . 812
5.3.34 SynGen . . . 815
GENCLS . . . 815
GENROU . . . 820
PLBVFU1 . . . 825
5.3.35 TimedEvent . . . 828
Toggle . . . 828
Fault . . . 829
Alter . . . 831
5.3.36 TurbineGov . . . 833
TG2 . . . 833
TGOV1 . . . 836
TGOV1DB . . . 840
TGOV1N . . . 843
TGOV1NDB . . . 847
IEEEG1 . . . 850
IEESGO . . . 855
GAST . . . 859
HYGOV . . . 862
HYGOVDB . . . 866
HYGOV4 . . . 870
5.3.37 Undefined . . . 874
5.3.38 VoltComp . . . 875
IEEEVC . . . 875
5.4 Config Reference . . . 877
5.4.1 System . . . 877
5.4.2 PFlow . . . 877
5.4.3 TDS . . . 878
5.4.4 EIG . . . 879
5.4.5 SE . . . 879
5.4.6 CPF . . . 879
5.5 API reference . . . 880
5.5.1 System . . . 880
andes.system . . . 880
andes.variables . . . 891
5.5.2 Routines . . . 897
andes.routines . . . 897

5.5.3 Plot . . . 916
andes.plot . . . 916
5.5.4 I/O . . . 919
andes.io . . . 920
5.5.5 Interoperability . . . 929
andes.interop . . . 929
5.5.6 Others . . . 938
andes.cli . . . 938
andes.main . . . 939
andes.utils.paths . . . 948
andes.utils.snapshot . . . 951
andes.utils.widgets . . . 952
5.6 Release notes . . . 953

6 Verification **985**
6.1 Overview . . . 985
6.2 Test Cases . . . 985
6.2.1 IEEE 14-Bus System . . . 985
6.2.2 NPCC System . . . 986
6.2.3 WECC System . . . 986
6.3 Verification Approach . . . 986
6.4 Detailed Results . . . 986
6.4.1 IEEE 14-Bus Verification . . . 986
Background . . . 986
Scenario 1: Line 1-2 trips at 1 sec. and reconnects after 2.0 sec. . . . 988
Scenario 2: Line 7-8 trips at 1 sec. and reconnects after 0.1 sec. . . . 989
Appendix: IEEE 14-Bus System Data . . . 991
6.4.2 CURENT NPCC Verification . . . 996
Background . . . 996
Scenario 1: Line 1-2 trips at 1 sec. and reconnects after 2.0 sec. . . . 997
Appendix: CURENT NPCC 140-Bus System Data . . . 998
6.4.3 CURENT WECC Verification . . . 1024
Background . . . 1024
Scenario 1: Line 38-39 trips and reconnects after 0.1 sec. . . . 1026
Scenario 2: Line 4-16 trips and reconnects after 0.1 sec. . . . 1028
Appendix: CURENT WECC 179-Bus System Data . . . 1030

Python Module Index **1067**

CHAPTER

ONE

ABOUT ANDES

ANDES is an open-source Python library for power system modeling, computation, analysis, and control. It serves as the dynamic simulation engine for the CURENT Large Scale Testbed (LTB), supporting power flow calculation, transient stability simulation, and small-signal stability analysis for transmission systems.

1.1 Symbolic-Numeric Framework

At its core, ANDES implements a hybrid symbolic-numeric framework that separates model description from numerical implementation. Device models are written as symbolic equations in Python, which ANDES processes using SymPy to generate optimized numerical code automatically. This allows researchers to prototype new models without manually deriving Jacobians or writing simulation code. The generated code is cached and reused, so the symbolic overhead is incurred only once.

1.2 What's Included

- **Comprehensive model library** — synchronous generators, exciters, turbine governors, PSS, and full second-generation renewable models (solar PV, Type 3 and Type 4 wind) following WECC specifications
- **Verified results** — models produce identical time-domain simulation results against commercial software for IEEE 14-bus and NPCC systems with GENROU and multiple controllers
- **Industry file formats** — PSS/E raw and dyr parsing for direct use of standard case files
- **Production performance** — 20-second transient simulation of a 2000-bus system completes in seconds on a typical desktop, achieved through vectorized NumPy operations and sparse matrix handling

1.3 Citation

If ANDES is used in your research, please cite:

> H. Cui, F. Li and K. Tomsovic, "Hybrid Symbolic-Numeric Framework for Power System Modeling and Analysis," *IEEE Transactions on Power Systems*, vol. 36, no. 2, pp. 1373-1384, March 2021. DOI: 10.1109/TPWRS.2020.3017019

1.4 Acknowledgments

This work was supported in part by the Engineering Research Center Program of the National Science Foundation and the Department of Energy under NSF Award Number EEC-1041877 and the CURENT Industry Partnership Program.

ANDES is developed and maintained by Hantao Cui. See the GitHub repository for a full list of contributors.

1.5 License

ANDES is released under the GNU General Public License v3.

CHAPTER

TWO

TUTORIALS

Step-by-step guides to learn ANDES, from installation to advanced analysis workflows.

These tutorials are designed to be followed in sequence for new users. Each tutorial builds on concepts from previous ones, gradually introducing more sophisticated analysis techniques. Experienced users can jump directly to topics of interest.

2.1 Learning Paths

2.1.1 New User

Start with tutorials 01-06 to learn the basics of ANDES:

- Installation and verification
- Running simulations
- Working with data files
- Visualizing results

2.1.2 Power System Analyst

After completing the basics, continue with tutorials 07-12 for advanced analysis:

- Small-signal stability assessment
- Parameter sensitivity studies
- Contingency analysis
- Dynamic control studies
- State estimation

2.1.3 Model Developer

For creating custom device models, see the *Modeling* section after completing these tutorials.

2.2 Installation

2.2.1 Quick Install

conda (recommended)

```
conda install -c conda-forge andes
```

pip

```
pip install andes
```

uv

```
uv pip install andes
```

2.2.2 New to Python

Setting Up miniforge

If you are new to Python and want to get started quickly, use miniforge, a conda-like package manager configured with conda-forge.

Step 1: Download the latest miniforge for your platform from miniforge releases.

- Most users: `x86_64(amd64)` for Intel and AMD processors
- Mac with Apple Silicon: `arm64(Apple Silicon)` for best performance

Complete the installation on your system.

Note: miniforge is a drop-in replacement for conda. If you have an existing conda installation, you can replace all `mamba` commands with `conda`.

If you are using Anaconda or Miniconda on Windows, open `Anaconda Prompt` instead of `Miniforge Prompt`.

Step 2: Open Terminal (Linux/macOS) or `Miniforge Prompt` (Windows, **not cmd!**).

You should see `(base)` prepended to the command prompt, e.g., `(base) C:\Users\username>`.

Create an environment for ANDES:

```
mamba create --name andes python=3.11
```

Activate the new environment:

```
mamba activate andes
```

Note: You will need to activate the `andes` environment every time in a new terminal session.

2.2.3 Using uv

uv is a fast Python package installer and resolver. If you're already familiar with uv, you can use it to install ANDES.

Install into current environment:

```
uv pip install andes
```

Create a new virtual environment with ANDES:

```
uv venv --python 3.12
source .venv/bin/activate  # On Windows: .venv\Scripts\activate
uv pip install andes
```

With extras:

```
uv pip install andes[dev]
```

Development install:

```
git clone https://github.com/curent/andes
cd andes
uv pip install -e .[dev]
```

Tip: For uv installation and detailed usage, see the uv documentation.

2.2.4 Extra Packages

Some ANDES features require extra packages not installed by default:

Group	Description
`dev`	Development packages (testing, documentation)
`interop`	Interoperability with other power system tools

Install extras with pip:

```
# Development packages
pip install andes[dev]

# All extras
pip install andes[all]
```

Note: Extra packages are not supported by conda/mamba installation. Use `pip` for extras.

2.2.5 Development Install

For users who want to modify code or develop new models. Changes to source code are reflected immediately without reinstallation.

Step 1: Clone the source code:

```
git clone https://github.com/curent/andes
```

Step 2: Install in development mode:

```
cd andes
pip install -e .
```

With extras:

```
pip install -e .[dev]
```

Note: ANDES uses `setuptools-scm` for versioning based on git tags. The version updates automatically when you `git pull` new changes.

Check version: `andes` or `python -c "import andes; print(andes.__version__)"`

2.2.6 Updating ANDES

Warning: If installed in development mode, use `git pull` to update. Do not run `conda install` or `pip install` as this creates duplicate installations.

conda/mamba:

```
conda install -c conda-forge --yes andes
```

pip:

```
pip install --upgrade andes
```

uv:

```
uv pip install --upgrade andes
```

Check *Release Notes* before updating for breaking changes.

2.2.7 Troubleshooting

Multiple Copies Installed

If you have both development and package installations, uninstall all copies:

```
conda remove andes
pip uninstall andes
```

Run both commands multiple times until neither finds the package.

Windows DLL Error

If you see:

```
ImportError: DLL load failed: The specified module could not be found.
```

This is a Windows Python path issue. The easiest fix is to install ANDES in a Conda/miniforge environment.

2.2.8 Next Steps

With ANDES installed, you're ready to run your first simulation:

- *First Simulation* - Load a test case and run power flow and time-domain simulation
- *Power Flow Analysis* - Deep dive into power flow analysis
- *Command Line Interface* - Command-line interface reference

2.3 First Simulation

This tutorial walks you through your first power system simulation using ANDES. By the end, you will have run power flow, time-domain simulation, and eigenvalue analysis on a test system. The entire workflow can be completed in about 5 minutes.

2.3.1 Import ANDES

Like other Python libraries, ANDES needs to be imported into an interactive Python environment before use. The `config_logger` function sets up the logging system that controls what messages ANDES displays during execution. Setting `stream_level=20` corresponds to the INFO level, which shows normal operation messages without excessive detail.

```
# display plots in notebooks
%matplotlib inline

import andes

andes.config_logger(stream_level=20)
```

2.3.2 Load a Test Case

ANDES includes several built-in test cases that are useful for learning and validation. The `andes.get_case()` function returns the full path to a built-in case file, which can then be passed to `andes.load()` to create a System object.

In this tutorial, we use the Kundur two-area test system, a well-known benchmark for transient stability studies. This system contains four generators, two areas connected by weak tie-lines, and includes dynamic models for exciters and governors.

```
ss = andes.load(andes.get_case('kundur/kundur_full.xlsx'))
```

```
Working directory: "/home/docs/checkouts/readthedocs.org/user_builds/andes/
↪checkouts/stable/docs/source/tutorials"
> Loaded generated Python code in "/home/docs/.andes/pycode".
Parsing input file "/home/docs/checkouts/readthedocs.org/user_builds/andes/
↪envs/stable/lib/python3.11/site-packages/andes/cases/kundur/kundur_full.xlsx
↪"...
Input file parsed in 0.0314 seconds.
Connectivity check completed in 0.0001 seconds.
-> System connectivity check results:
  No islanded bus detected.
  System is interconnected.
  Each island has a slack bus correctly defined and enabled.
System internal structure set up in 0.0209 seconds.
```

The variable `ss` is now an instance of the `System` class, which serves as the central container for all power system data, device models, and analysis routines. The naming convention `ss` (short for "system") is commonly used throughout ANDES examples and documentation. You can inspect the system by simply typing `ss` to see a summary of loaded models.

```
ss
```

```
<andes.system.facade.System at 0x774e122ebb10>
```

2.3.3 Run Power Flow

Before performing any dynamic simulation, you must first solve the power flow to establish a valid steady-state operating point. Power flow calculates the voltage magnitude and angle at each bus, as well as the active and reactive power flows throughout the network. ANDES uses the Newton-Raphson method, which typically converges within a few iterations for well-conditioned systems.

The power flow routine is accessed through `ss.PFlow`, and you run it by calling the `run()` method.

Note: In the PDF version of this documentation, detailed solver outputs are hidden to reduce repetition. To see the full outputs, refer to the online HTML documentation.

```
ss.PFlow.run()
```

```
-> Power flow calculation
           Numba: Off
   Sparse solver: KLU
 Solution method: NR method
Power flow initialized in 0.0039 seconds.
0: |F(x)| = 14.9282832
1: |F(x)| = 3.608627841
2: |F(x)| = 0.1701107882
3: |F(x)| = 0.002038626956
4: |F(x)| = 3.745103977e-07
Converged in 5 iterations in 0.0026 seconds.
Report saved to "kundur_full_out.txt" in 0.0009 seconds.
```

```
True
```

After the power flow converges, you can access the solved values through the model objects. For example, the bus voltage magnitudes are stored in `ss.Bus.v.v`, where the first `v` refers to the voltage variable and the second `v` accesses its numerical value array.

```
ss.Bus.v.v
```

```
array([1.        , 1.        , 1.        , 1.        , 0.98337472,
       0.96908586, 0.95621812, 0.95400019, 0.96856366, 0.98377143])
```

2.3.4 Run Time-Domain Simulation

Time-domain simulation (TDS) solves the differential-algebraic equations (DAEs) that govern power system dynamics. This allows you to study how the system responds to disturbances such as faults, line trips, or generator outages. ANDES uses the implicit trapezoidal method for numerical integration, which provides good accuracy and numerical stability.

Before running TDS, you can configure simulation parameters through `ss.TDS.config`. The most commonly adjusted parameter is `tf`, which sets the simulation end time in seconds. The Kundur test case includes a pre-defined line trip disturbance at t=2 seconds, so we simulate for 10 seconds to observe the system response.

```
ss.TDS.config.tf = 10
ss.TDS.run()
```

```
DAE compaction: removed 5 algebraic variable slots (m: 149 -> 144)
Initialization for dynamics completed in 0.0162 seconds.
Initialization was successful.
<Toggle 1>: Line.Line_8 status changed to 0 at t=2.0 sec.
Simulation to t=10.00 sec completed in 0.5468 seconds.
Outputs to "kundur_full_out.lst" and "kundur_full_out.npz".
Outputs written in 0.0144 seconds.
```

```
  0%|          | 0/100 [00:00<?, ?%/s]
```

```
True
```

The simulation output shows the progress as time advances. When the line trips at t=2s, the generators experience a sudden change in loading conditions, causing their rotors to oscillate as they seek a new equilibrium point.

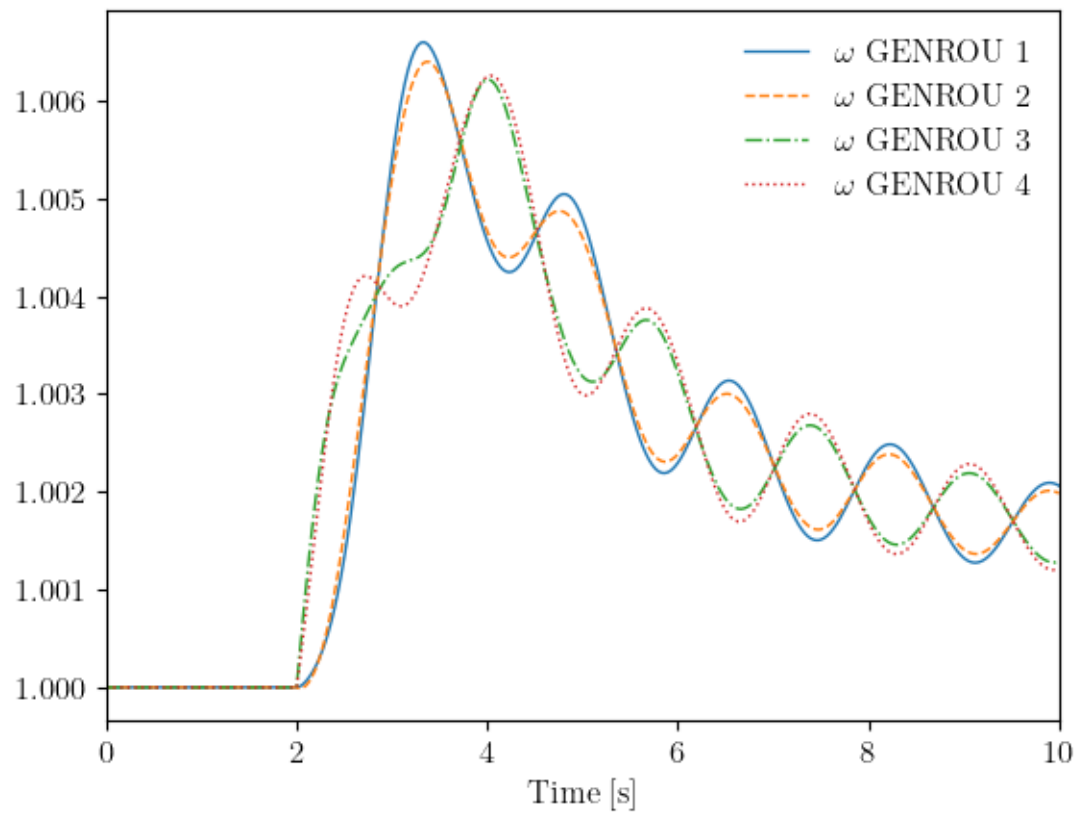
2.3.5 Plot Results

ANDES provides a built-in plotting interface through `ss.TDS.plt` that makes it easy to visualize simulation results. The `plot()` method accepts a variable object and automatically retrieves its time series data.

To see the dynamic response, we plot the generator rotor speeds (`omega`). In a stable system, the speeds should oscillate after the disturbance and eventually settle to a new steady-state value. The GENROU model represents round-rotor synchronous generators, which are the four machines in this test system.

```
ss.TDS.plt.plot(ss.GENROU.omega)
```

```
Using LaTeX for rendering. If an error occurs:
a) If you are using `andes plot`, disable with option "-d",
b) If you are using `plot()`, set "latex=False".
```




```
(<Figure size 640x480 with 1 Axes>, <Axes: xlabel='Time [s]'>)
```

The plot shows the characteristic inter-area oscillations of the Kundur two-area system. Generators in the same area tend to swing together, while generators in different areas swing against each other. This oscillatory mode is a key concern in power system stability analysis.

2.3.6 Run Eigenvalue Analysis

Eigenvalue analysis examines small-signal stability by linearizing the system equations around the operating point and computing the eigenvalues of the state matrix. The location of eigenvalues in the complex plane indicates system stability: eigenvalues with negative real parts correspond to stable modes, while those with positive real parts indicate instability.

For eigenvalue analysis, we load a fresh system instance because TDS modifies internal states during simulation. After running power flow to establish the operating point, we call `ss2.EIG.run()` to perform the eigenvalue calculation.

```
ss2 = andes.load(andes.get_case('kundur/kundur_full.xlsx'))
ss2.PFlow.run()
ss2.EIG.run()
```

```
Working directory: "/home/docs/checkouts/readthedocs.org/user_builds/andes/
↪checkouts/stable/docs/source/tutorials"
```

(continues on next page)

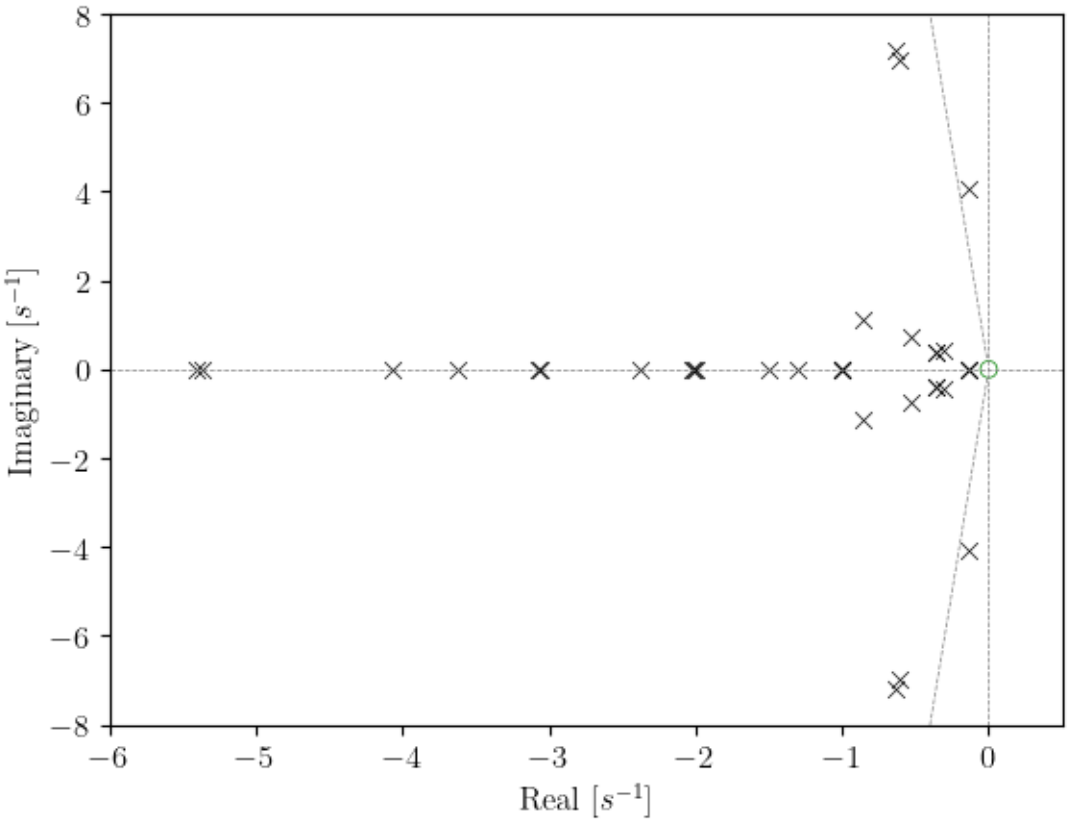
(continued from previous page)

```
> Reloaded generated Python code of module "pycode".
Parsing input file "/home/docs/checkouts/readthedocs.org/user_builds/andes/
↪envs/stable/lib/python3.11/site-packages/andes/cases/kundur/kundur_full.xlsx
↪"...
Input file parsed in 0.0297 seconds.
Connectivity check completed in 0.0001 seconds.
-> System connectivity check results:
  No islanded bus detected.
  System is interconnected.
  Each island has a slack bus correctly defined and enabled.
System internal structure set up in 0.0194 seconds.

-> Power flow calculation
           Numba: Off
   Sparse solver: KLU
 Solution method: NR method
Power flow initialized in 0.0030 seconds.
0: |F(x)| = 14.9282832
1: |F(x)| = 3.608627841
2: |F(x)| = 0.1701107882
3: |F(x)| = 0.002038626956
4: |F(x)| = 3.745103977e-07
Converged in 5 iterations in 0.0024 seconds.
Report saved to "kundur_full_out.txt" in 0.0005 seconds.
DAE compaction: removed 5 algebraic variable slots (m: 149 -> 144)
Initialization for dynamics completed in 0.0151 seconds.
Initialization was successful.

-> Eigenvalue Analysis:
  Positive       0
  Zeros          1
  Negative      52
Eigenvalue analysis finished in 0.0026 seconds.
Report saved to "kundur_full_eig.txt".
```

```
True
```

The eigenvalue results can be visualized on the s-plane using the `plot()` method. Each point represents an eigenvalue, with the x-axis showing the real part (damping) and the y-axis showing the imaginary part (oscillation frequency).

```
ss2.EIG.plot()
```




```
(<Figure size 640x480 with 1 Axes>,
 <Axes: xlabel='Real [$s^{-1}$]', ylabel='Imaginary [$s^{-1}$]'>)
```

2.3.7 Key Concepts

The following table summarizes the main components you have encountered in this tutorial.

Concept	Description
`System`	Central container that holds all models, parameters, and analysis routines
`PFlow`	Power flow routine using Newton-Raphson iteration
`TDS`	Time-domain simulation using implicit trapezoidal integration
`EIG`	Eigenvalue analysis for small-signal stability assessment
Models	Device representations such as `Bus`, `PQ`, `GENROU`, `TGOV1`, `EXDC2`

2.3.8 Using the Command Line

In addition to the Python interface, ANDES provides a command-line interface (CLI) for running simulations without writing code. This is convenient for batch processing or quick analyses. The following examples demonstrate common CLI usage patterns.

```
# Run power flow (default routine)
andes run ieee14.json

# Run time-domain simulation
andes run kundur_full.xlsx -r tds

# Run eigenvalue analysis
andes run kundur_full.xlsx -r eig

# Plot results from output files
andes plot kundur_full_out.lst 0 5 6 7 8
```

The `-r` flag specifies which routine to run. Without it, ANDES runs power flow by default. Output files are automatically generated with the case name as a prefix.

2.3.9 Cleanup

Simulations generate output files including result data (`.npz`), variable listings (`.lst`), and reports (`.txt`). The following command removes these files from the current directory.

```
!andes misc -C
```

```
"/home/docs/checkouts/readthedocs.org/user_builds/andes/checkouts/stable/docs/
↪source/tutorials/kundur_full_eig.txt" removed.
"/home/docs/checkouts/readthedocs.org/user_builds/andes/checkouts/stable/docs/
↪source/tutorials/kundur_full_out.txt" removed.
"/home/docs/checkouts/readthedocs.org/user_builds/andes/checkouts/stable/docs/
↪source/tutorials/kundur_full_out.lst" removed.
"/home/docs/checkouts/readthedocs.org/user_builds/andes/checkouts/stable/docs/
↪source/tutorials/kundur_full_out.npz" removed.
```

2.3.10 Next Steps

Now that you have completed your first simulation, you can explore more advanced topics:

- *Power Flow Analysis* - Detailed power flow analysis and troubleshooting
- *Time-Domain Simulation* - Adding disturbances and configuring simulations
- *Data and File Formats* - Loading different file formats and modifying parameters

2.4 Power Flow Analysis

Power flow analysis, also known as load flow, is the foundation of power system studies. It calculates the steady-state operating point of a power system by solving a set of nonlinear algebraic equations that represent power balance at each bus. The results include voltage magnitudes and angles at all buses, as well as active and reactive power flows on all branches.

This tutorial covers running power flow in ANDES, accessing and interpreting results, configuring solver options, and troubleshooting convergence issues.

2.4.1 Setup

We begin by importing ANDES and configuring the logger. The logger controls what information is displayed during execution, with level 20 (INFO) being appropriate for normal use.

```
%matplotlib inline

import andes
andes.config_logger(stream_level=20)
```

2.4.2 Running Power Flow

To run power flow, you first load a case file to create a System object, then call the `run()` method on the `PFlow` routine. ANDES implements the Newton-Raphson method, which iteratively solves the power balance equations until the mismatch (residual) falls below the convergence tolerance.

For this tutorial, we use the IEEE 14-bus test system, a standard benchmark that contains 14 buses, 5 generators, and 20 branches. This system is small enough to examine results easily while being realistic enough to demonstrate typical power flow behavior.

```
ss = andes.load(andes.get_case('ieee14/ieee14.json'))
ss.PFlow.run()
```

```
Working directory: "/home/docs/checkouts/readthedocs.org/user_builds/andes/
↪checkouts/stable/docs/source/tutorials"
> Loaded generated Python code in "/home/docs/.andes/pycode".
Parsing input file "/home/docs/checkouts/readthedocs.org/user_builds/andes/
↪envs/stable/lib/python3.11/site-packages/andes/cases/ieee14/ieee14.json"...
Input file parsed in 0.0021 seconds.
Connectivity check completed in 0.0001 seconds.
-> System connectivity check results:
  No islanded bus detected.
  System is interconnected.
  Each island has a slack bus correctly defined and enabled.
System internal structure set up in 0.0200 seconds.

-> Power flow calculation
           Numba: Off
```

(continues on next page)

(continued from previous page)

```
   Sparse solver: KLU
 Solution method: NR method
Power flow initialized in 0.0034 seconds.
0: |F(x)| = 0.5605182134
1: |F(x)| = 0.006202200332
2: |F(x)| = 5.819382827e-06
3: |F(x)| = 6.957087684e-12
Converged in 4 iterations in 0.0022 seconds.
Report saved to "ieee14_out.txt" in 0.0011 seconds.
```

```
True
```

After the solver completes, you can check whether the solution converged and how many iterations were required. A well-conditioned power flow typically converges in 3-6 iterations. If the iteration count is high or convergence fails, see the Troubleshooting section below.

```
print(f"Converged: {ss.PFlow.converged}")
print(f"Iterations: {ss.PFlow.niter}")
```

```
Converged: True
Iterations: 3
```

Command Line Alternative

Power flow can also be executed from the command line without writing Python code. This is useful for quick analyses or batch processing. The following command runs power flow on a case file:

```
andes run ieee14.json
```

The CLI automatically generates output files containing the results.

2.4.3 Accessing Results

After power flow converges, results are stored in the model objects that make up the System. Each model (such as `Bus`, `Line`, `PV`) contains variables whose solved values can be accessed through the `.v` attribute. Understanding how to navigate this structure is essential for extracting the information you need.

Bus Results

Bus voltages are the primary output of power flow analysis. The voltage magnitude is stored in `ss.Bus.v` and the voltage angle in `ss.Bus.a`. To access the numerical values as NumPy arrays, append `.v` to get the value array.

```
# Voltage magnitudes (per unit)
ss.Bus.v.v
```

```
array([1.03      , 1.03      , 1.01      , 1.01140345, 1.0172555 ,
       1.03      , 1.0224715 , 1.03      , 1.02176879, 1.01554206,
       1.01911514, 1.01740698, 1.01445023, 1.0163402 ])
```

```
# Voltage angles (radians)
ss.Bus.a.v
```

```
array([ 7.95343065e-14, -3.07888354e-02, -6.17345595e-02, -7.69651166e-02,
       -6.70734614e-02, -1.12621496e-01, -8.52626931e-02, -2.68773277e-02,
       -1.26464055e-01, -1.29424839e-01, -1.23564069e-01, -1.30428971e-01,
       -1.34752637e-01, -1.65476685e-01])
```

For a more comprehensive view, you can retrieve all bus parameters and variables as a pandas DataFrame using `as_df()`. This is particularly useful for inspecting multiple quantities at once or exporting results to other tools.

```
ss.Bus.as_df()
```

```
     idx    u   name     Vn  vmax  vmin       v0        a0  xcoord  ycoord  \
uid
0      1  1.0   BUS1   69.0   1.1   0.9  1.03000  0.000000       0       0
1      2  1.0   BUS2   69.0   1.1   0.9  1.01970 -0.027981       0       0
2      3  1.0   BUS3   69.0   1.1   0.9  1.00042 -0.060097       0       0
3      4  1.0   BUS4   69.0   1.1   0.9  0.99858 -0.074721       0       0
4      5  1.0   BUS5   69.0   1.1   0.9  1.00443 -0.064315       0       0
5      6  1.0   BUS6  138.0   1.1   0.9  0.99871 -0.109998       0       0
6      7  1.0   BUS7  138.0   1.1   0.9  1.00682 -0.084285       0       0
7      8  1.0   BUS8   69.0   1.1   0.9  1.01895 -0.024339       0       0
8      9  1.0   BUS9  138.0   1.1   0.9  1.00193 -0.127502       0       0
9     10  1.0  BUS10  138.0   1.1   0.9  0.99351 -0.130202       0       0
10    11  1.0  BUS11  138.0   1.1   0.9  0.99245 -0.122948       0       0
11    12  1.0  BUS12  138.0   1.1   0.9  0.98639 -0.128934       0       0
12    13  1.0  BUS13  138.0   1.1   0.9  0.98403 -0.133786       0       0
13    14  1.0  BUS14  138.0   1.1   0.9  0.99063 -0.166916       0       0

     area  zone  owner
uid
0       1     1      1
1       1     1      1
2       1     1      1
3       1     1      1
4       1     1      1
5       2     2      2
6       2     2      2
7       2     2      2
8       2     2      2
9       2     2      2
10      2     2      2
11      2     2      2
12      2     2      2
13      2     2      2
```

Generator Results

Generator outputs are distributed across different model types depending on their control mode. In power flow, `PV` buses represent generators with fixed voltage magnitude and active power output, while `Slack` (or swing) buses balance the system by absorbing any mismatch. The solved reactive power output `q` is determined by the power flow solution.

```
print("PV generators - Active power (pu):", ss.PV.p.v)
print("Slack generator - Active power (pu):", ss.Slack.p.v)
```

```
PV generators - Active power (pu): [0.4  0.4  0.3  0.35]
Slack generator - Active power (pu): [0.81427214]
```

```
print("PV generators - Reactive power (pu):", ss.PV.q.v)
print("Slack generator - Reactive power (pu):", ss.Slack.q.v)
```

```
PV generators - Reactive power (pu): [0.30436147 0.12597133 0.20986596 0.
↪07396392]
Slack generator - Reactive power (pu): [-0.21617103]
```

Line Flows

Branch power flows indicate how power is transmitted through the network. For each line, `a1` represents the active power flow at the from-bus end, and `a2` represents the active power flow at the to-bus end. The difference between these values accounts for transmission losses.

```
# Show flows for first 5 lines
print("From-bus active power (pu):", ss.Line.a1.v[:5])
print("To-bus active power (pu):", ss.Line.a2.v[:5])
```

```
From-bus active power (pu): [ 7.95343065e-14  7.95343065e-14 -3.07888354e-02 -
↪3.07888354e-02
 -3.07888354e-02]
To-bus active power (pu): [-0.03078884 -0.06707346 -0.06173456 -0.07696512 -0.
↪06707346]
```

Exploring Bus Connections

When working with an unfamiliar system, you often need to know what devices are connected to a particular bus. The `find_connected()` method scans all models and returns the devices referencing a given bus (or any other device). This eliminates the need to manually search through model data.

```
# What's connected to Bus 1 (slack bus)?
ss.find_connected('Bus', 1)
```

```
OrderedDict([('Slack', [1]),
             ('Line', ['Line_1', 'Line_2']),
             ('GENROU', ['GENROU_1']),
             ('BusFreq', ['BusFreq_2'])])
```

2.4.4 Configuration

The power flow solver behavior can be customized through configuration options accessible via `ss.PFlow.config`. These settings control convergence criteria, iteration limits, and solver behavior. Viewing the current configuration shows all available options and their values.

```
ss.PFlow.config
```

```
OrderedDict([('linsolve', 0),
             ('tol', 1e-06),
             ('max_iter', 25),
             ('method', 'NR'),
             ('n_factorize', 4),
             ('report', 1),
             ('degree', 0),
             ('init_tds', 0),
             ('linesearch', 1)])
```

The most commonly adjusted options are summarized below:

Option	Default	Description
`max_iter`	25	Maximum number of Newton-Raphson iterations
`tol`	1e-6	Convergence tolerance for power mismatch
`method`	'NR'	Solution method (NR = Newton-Raphson)
`init_tds`	0	Whether to initialize TDS models after power flow

To modify these settings, assign new values directly to the config attributes. Changes take effect on the next `run()` call.

```
ss.PFlow.config.max_iter = 50
ss.PFlow.config.tol = 1e-8
```

For persistent configuration that applies to all ANDES sessions, you can edit the configuration file located at `~/.andes/andes.conf`. Settings in this file are loaded automatically when ANDES starts.

```
[PFlow]
max_iter = 50
tol = 1e-8
```

2.4.5 Troubleshooting

Non-Convergence

Power flow convergence failures are typically caused by one of three issues:

1. **Data errors**: Zero impedance lines create singular Jacobian matrices, and isolated buses have no valid solution. Check your network topology for disconnected components.
2. **Infeasible operating point**: If total load exceeds generation capacity, or if reactive power requirements cannot be met by available sources, no solution exists. Verify that generation and load are balanced.
3. **Poor initial guess**: The Newton-Raphson method requires a reasonable starting point. For very stressed systems, the flat start (1.0 pu voltage, 0 angle) may be too far from the solution.

When convergence fails, try the following debugging steps:

```
# Allow more iterations for difficult cases
ss.PFlow.config.max_iter = 100

# Enable PV-to-PQ conversion when generators hit reactive limits
ss.PV.config.pv2pq = 1
```

Viewing Iteration Progress

To diagnose convergence issues, you can enable debug-level logging which displays the residual magnitude at each iteration. A healthy convergence shows the residual decreasing rapidly (quadratic convergence is characteristic of Newton-Raphson). If the residual oscillates or decreases slowly, there may be numerical issues with the case data.

```
andes.config_logger(stream_level=10)  # DEBUG level

ss_debug = andes.load(andes.get_case('ieee14/ieee14.json'))
ss_debug.PFlow.run()
```

```
Working directory: "/home/docs/checkouts/readthedocs.org/user_builds/andes/
↪checkouts/stable/docs/source/tutorials"
Found files: ['/home/docs/checkouts/readthedocs.org/user_builds/andes/envs/
↪stable/lib/python3.11/site-packages/andes/cases/ieee14/ieee14.json']
Registered deprecated group alias ACTopology -> ACNode
> Reloaded generated Python code of module "pycode".
Input format guessed as json.
Parsing input file "/home/docs/checkouts/readthedocs.org/user_builds/andes/
↪envs/stable/lib/python3.11/site-packages/andes/cases/ieee14/ieee14.json"...
Input file parsed in 0.0022 seconds.
Setting internal address for Bus
Setting internal address for PQ
Setting internal address for PV
Setting internal address for Slack
Setting internal address for Shunt
Setting internal address for Line
```

(continues on next page)

(continued from previous page)

```
Setting internal address for Area
Entering connectivity check.
Connectivity check completed in 0.0002 seconds.
-> System connectivity check results:
  No islanded bus detected.
  System is interconnected.
  Bus indices in interconnected system (0-based): [[0, 4, 5, 12, 13, 8, 9, 6,␣
↪7, 11, 10, 3, 2, 1]]
  Each island has a slack bus correctly defined and enabled.
System internal structure set up in 0.0207 seconds.

-> Power flow calculation
           Numba: Off
   Sparse solver: KLU
 Solution method: NR method
========== Bus has <pflow_init> = True ==========
Initialization sequence:
a -> v
Bus: Pass 1 — unconstrained
Bus: Pass 2 — with discrete evaluation
========== PQ has <pflow_init> = True ==========
Initialization sequence:
a -> v
PQ: Pass 1 — unconstrained
PQ: Pass 2 — with discrete evaluation
========== PV has <pflow_init> = True ==========
Initialization sequence:
q -> a -> v
PV: Pass 1 — unconstrained
PV: Pass 2 — with discrete evaluation
========== Slack has <pflow_init> = True ==========
Initialization sequence:
q -> p -> a -> v
Slack: Pass 1 — unconstrained
Slack: Pass 2 — with discrete evaluation
========== Shunt has <pflow_init> = True ==========
Initialization sequence:
a -> v
Shunt: Pass 1 — unconstrained
Shunt: Pass 2 — with discrete evaluation
========== Line has <pflow_init> = True ==========
Initialization sequence:
a1 -> a2 -> v1 -> v2
Line: Pass 1 — unconstrained
Line: Pass 2 — with discrete evaluation
========== Area has <pflow_init> = True ==========
Initialization sequence:

Area: Pass 1 — unconstrained
Area: Pass 2 — with discrete evaluation
Power flow initialized in 0.0041 seconds.
Max. algeb mismatch 0.5605182134 on v Bus 6
```

(continues on next page)

(continued from previous page)

```
0: |F(x)| = 0.5605182134
Jacobian updated at t=-1.000000.
Max. algeb mismatch 0.006202200332 on v Bus 13
1: |F(x)| = 0.006202200332
Jacobian updated at t=-1.000000.
Max. algeb mismatch 5.819382827e-06 on v Bus 13
2: |F(x)| = 5.819382827e-06
Jacobian updated at t=-1.000000.
Max. algeb mismatch 6.957087684e-12 on v Bus 5
3: |F(x)| = 6.957087684e-12
Converged in 4 iterations in 0.0023 seconds.
Report saved to "ieee14_out.txt" in 0.0016 seconds.
```

```
True
```

The output shows the residual norm at each iteration:

```
0: |F(x)| = 14.9282832
1: |F(x)| = 3.608627841
2: |F(x)| = 0.1701107882
...
```

Quadratic convergence means that each iteration roughly squares the number of correct digits, so you should see the residual drop by several orders of magnitude in just a few iterations.

2.4.6 Power Flow Report

After a successful power flow, ANDES generates a text report file containing formatted results. This report includes system statistics, bus data (voltage, angle, injections), and branch data (flows, losses). The report path can be accessed through the `files` attribute.

```
ss.files.txt
```

```
'ieee14_out.txt'
```

The report is organized into sections:

1. **System statistics**: Total generation, load, and losses
2. **Bus data**: Voltage magnitude and angle, P/Q injection at each bus
3. **Line data**: Active and reactive power flows, losses on each branch
4. **Other algebraic variables**: Additional solved quantities from device models

2.4.7 Example: Formatted Bus Voltage Display

The following example demonstrates how to iterate through bus results and display them in a readable format. This pattern is useful for creating custom reports or validating results against other tools.

```
import math

print("Bus Voltage Results")
print("-" * 40)
for name, v, a in zip(ss.Bus.name.v, ss.Bus.v.v, ss.Bus.a.v):
    print(f"Bus {name:>4}: V = {v:.4f} pu, θ = {math.degrees(a):>7.2f}°")
```

```
Bus Voltage Results
----------------------------------------
Bus BUS1: V = 1.0300 pu, θ =    0.00°
Bus BUS2: V = 1.0300 pu, θ =   -1.76°
Bus BUS3: V = 1.0100 pu, θ =   -3.54°
Bus BUS4: V = 1.0114 pu, θ =   -4.41°
Bus BUS5: V = 1.0173 pu, θ =   -3.84°
Bus BUS6: V = 1.0300 pu, θ =   -6.45°
Bus BUS7: V = 1.0225 pu, θ =   -4.89°
Bus BUS8: V = 1.0300 pu, θ =   -1.54°
Bus BUS9: V = 1.0218 pu, θ =   -7.25°
Bus BUS10: V = 1.0155 pu, θ =   -7.42°
Bus BUS11: V = 1.0191 pu, θ =   -7.08°
Bus BUS12: V = 1.0174 pu, θ =   -7.47°
Bus BUS13: V = 1.0145 pu, θ =   -7.72°
Bus BUS14: V = 1.0163 pu, θ =   -9.48°
```

2.4.8 Cleanup

Remove output files generated during this tutorial.

```
!andes misc -C
```

```
"/home/docs/checkouts/readthedocs.org/user_builds/andes/checkouts/stable/docs/
↪source/tutorials/ieee14_out.txt" removed.
```

2.4.9 Next Steps

With power flow complete, you can proceed to dynamic analysis:

- *Time-Domain Simulation* - Time-domain simulation with disturbances
- *Data and File Formats* - Loading different file formats and modifying parameters

2.5 Time-Domain Simulation

Time-domain simulation (TDS) solves the differential-algebraic equations (DAEs) that govern power system dynamics. This type of analysis is essential for studying transient stability, which determines whether a power system can maintain synchronism following disturbances such as faults, line trips, or generator outages.

ANDES uses the implicit trapezoidal method for numerical integration, which provides a good balance between accuracy and computational efficiency. The method is A-stable, meaning it handles stiff systems well without requiring excessively small time steps.

This tutorial covers running time-domain simulations, adding various types of disturbances, configuring simulation parameters, accessing results, and troubleshooting common issues.

2.5.1 Setup

```
%matplotlib inline

import andes
andes.config_logger(stream_level=20)
```

2.5.2 Basic Simulation Workflow

A time-domain simulation always begins with a converged power flow solution, which provides the initial steady-state operating point. From this equilibrium, the DAE solver advances time while tracking how system variables evolve in response to any disturbances.

The basic workflow consists of three steps: load the case, run power flow, then run TDS. The simulation end time is controlled by `ss.TDS.config.tf`.

```
ss = andes.load(andes.get_case('kundur/kundur_full.xlsx'))
ss.PFlow.run()

ss.TDS.config.tf = 10  # Simulate for 10 seconds
ss.TDS.run()
```

```
Working directory: "/home/docs/checkouts/readthedocs.org/user_builds/andes/
↪checkouts/stable/docs/source/tutorials"
> Loaded generated Python code in "/home/docs/.andes/pycode".
Parsing input file "/home/docs/checkouts/readthedocs.org/user_builds/andes/
↪envs/stable/lib/python3.11/site-packages/andes/cases/kundur/kundur_full.xlsx
↪"...
Input file parsed in 0.0315 seconds.
Connectivity check completed in 0.0001 seconds.
-> System connectivity check results:
  No islanded bus detected.
  System is interconnected.
  Each island has a slack bus correctly defined and enabled.
System internal structure set up in 0.0203 seconds.
```

(continues on next page)

(continued from previous page)

```
-> Power flow calculation
           Numba: Off
   Sparse solver: KLU
 Solution method: NR method
Power flow initialized in 0.0031 seconds.
0: |F(x)| = 14.9282832
1: |F(x)| = 3.608627841
2: |F(x)| = 0.1701107882
3: |F(x)| = 0.002038626956
4: |F(x)| = 3.745103977e-07
Converged in 5 iterations in 0.0025 seconds.
Report saved to "kundur_full_out.txt" in 0.0008 seconds.
DAE compaction: removed 5 algebraic variable slots (m: 149 -> 144)
Initialization for dynamics completed in 0.0151 seconds.
Initialization was successful.
<Toggle 1>: Line.Line_8 status changed to 0 at t=2.0 sec.
Simulation to t=10.00 sec completed in 0.5416 seconds.
Outputs to "kundur_full_out.lst" and "kundur_full_out.npz".
Outputs written in 0.0142 seconds.
```

```
  0%|          | 0/100 [00:00<?, ?%/s]
```

```
True
```

The Kundur test case includes a pre-defined line trip at t=2 seconds, which triggers electromechanical oscillations between the two areas. The progress bar shows simulation advancement, and the final message confirms successful completion.

Command Line Alternative

Time-domain simulation can be run from the command line using the `-r tds` flag. The `--tf` option sets the simulation end time.

```
andes run kundur_full.xlsx -r tds --tf 20
```

2.5.3 Adding Disturbances

Disturbances are essential for transient stability analysis because they perturb the system from its equilibrium and allow you to observe the dynamic response. ANDES provides three main disturbance devices:

Device	Purpose
`Fault`	Three-phase-to-ground bus fault
`Toggle`	Connect or disconnect any device
`Alter`	Change parameter values during simulation

To add disturbances, you must load the case with `setup=False`, add the disturbance devices, then call `setup()` before running. This sequence is necessary because the system structure is finalized during setup.

See also:

For details on how timed events are implemented internally using `TimerParam` callbacks, see *System Architecture*.

Three-Phase Faults

The `Fault` model applies a three-phase-to-ground fault on a bus. This is the most severe type of fault and is commonly used for stability screening. The fault is characterized by its location (bus), application time (`tf`), and clearing time (`tc`).

```
ss_fault = andes.load(andes.get_case('kundur/kundur_full.xlsx'), setup=False)

# Disable the existing Toggle to isolate the fault effect
ss_fault.Toggle.set('u', 1, 0)

# Add fault on bus 5: apply at t=1.0s, clear at t=1.1s
ss_fault.add('Fault', bus=5, tf=1.0, tc=1.1)

ss_fault.setup()
ss_fault.PFlow.run()
ss_fault.TDS.config.tf = 5
ss_fault.TDS.run()
```

```
Working directory: "/home/docs/checkouts/readthedocs.org/user_builds/andes/
↪checkouts/stable/docs/source/tutorials"
> Reloaded generated Python code of module "pycode".
Parsing input file "/home/docs/checkouts/readthedocs.org/user_builds/andes/
↪envs/stable/lib/python3.11/site-packages/andes/cases/kundur/kundur_full.xlsx
↪"...
Input file parsed in 0.0305 seconds.
Connectivity check completed in 0.0001 seconds.
-> System connectivity check results:
  No islanded bus detected.
  System is interconnected.
  Each island has a slack bus correctly defined and enabled.
System internal structure set up in 0.0195 seconds.

-> Power flow calculation
           Numba: Off
   Sparse solver: KLU
 Solution method: NR method
Power flow initialized in 0.0030 seconds.
0: |F(x)| = 14.9282832
1: |F(x)| = 3.608627841
2: |F(x)| = 0.1701107882
3: |F(x)| = 0.002038626956
4: |F(x)| = 3.745103977e-07
```

(continues on next page)

(continued from previous page)

```
Converged in 5 iterations in 0.0024 seconds.
Report saved to "kundur_full_out.txt" in 0.0006 seconds.
DAE compaction: removed 5 algebraic variable slots (m: 149 -> 144)
Initialization for dynamics completed in 0.0157 seconds.
Initialization was successful.
<Fault Fault_2>: Applying fault on Bus (idx=5) at t=1.0 sec.
<Fault Fault_2>: Clearing fault on Bus (idx=5) at t=1.1 sec.
Simulation to t=5.00 sec completed in 0.3518 seconds.
Outputs to "kundur_full_out.lst" and "kundur_full_out.npz".
Outputs written in 0.0074 seconds.
```

```
  0%|          | 0/100 [00:00<?, ?%/s]
```

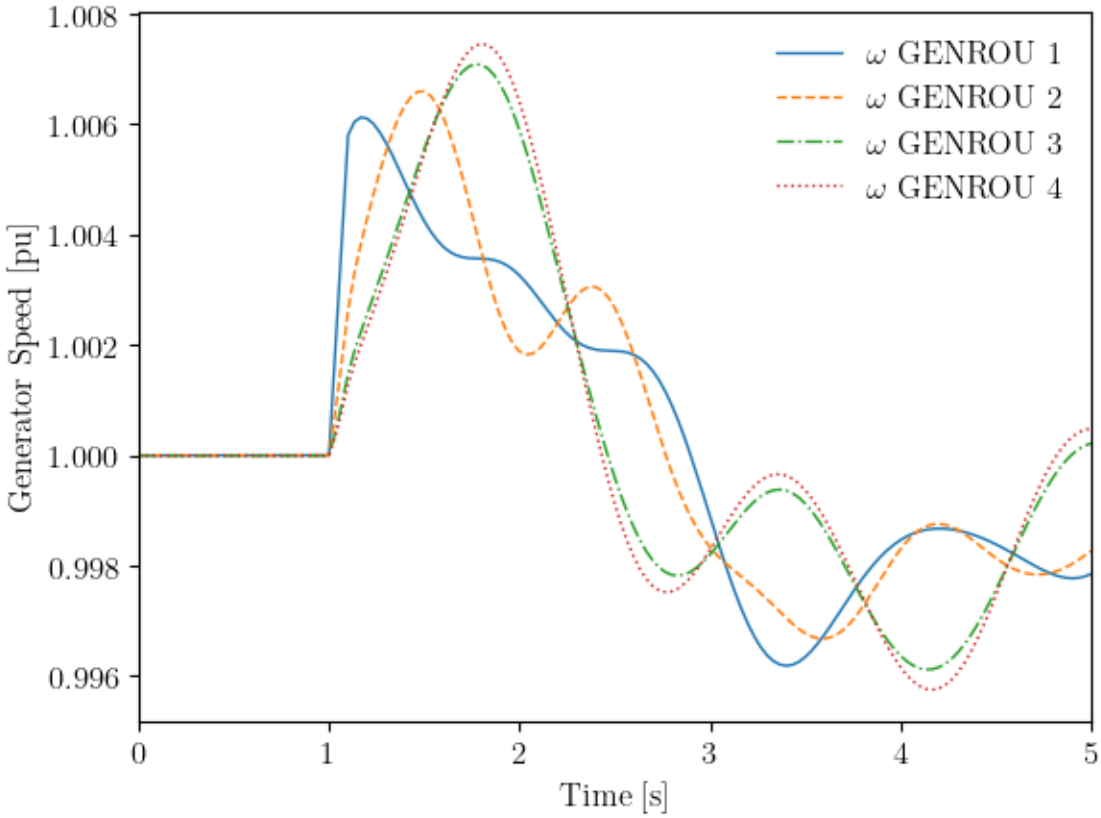
```
True
```

The fault parameters control its severity and duration:

Parameter	Description	Default
`bus`	Bus index for fault location	(required)
`tf`	Fault application time [s]	(required)
`tc`	Fault clearing time [s]	(required)
`rf`	Fault resistance [pu]	0
`xf`	Fault reactance [pu]	1e-6

For a permanent fault that is never cleared, set `tc` to a time beyond the simulation end.

```
ss_fault.TDS.plt.plot(ss_fault.GENROU.omega, ylabel='Generator Speed [pu]')
```




```
(<Figure size 640x480 with 1 Axes>,
 <Axes: xlabel='Time [s]', ylabel='Generator Speed [pu]'>)
```

Line and Device Trips (Toggle)

The `Toggle` model changes the connectivity status of any device at a specified time. This is useful for simulating line trips, generator trips, load shedding, or breaker operations. Unlike faults which modify the network admittance, toggles completely remove or reconnect devices.

The following example demonstrates a line trip followed by reclosure.

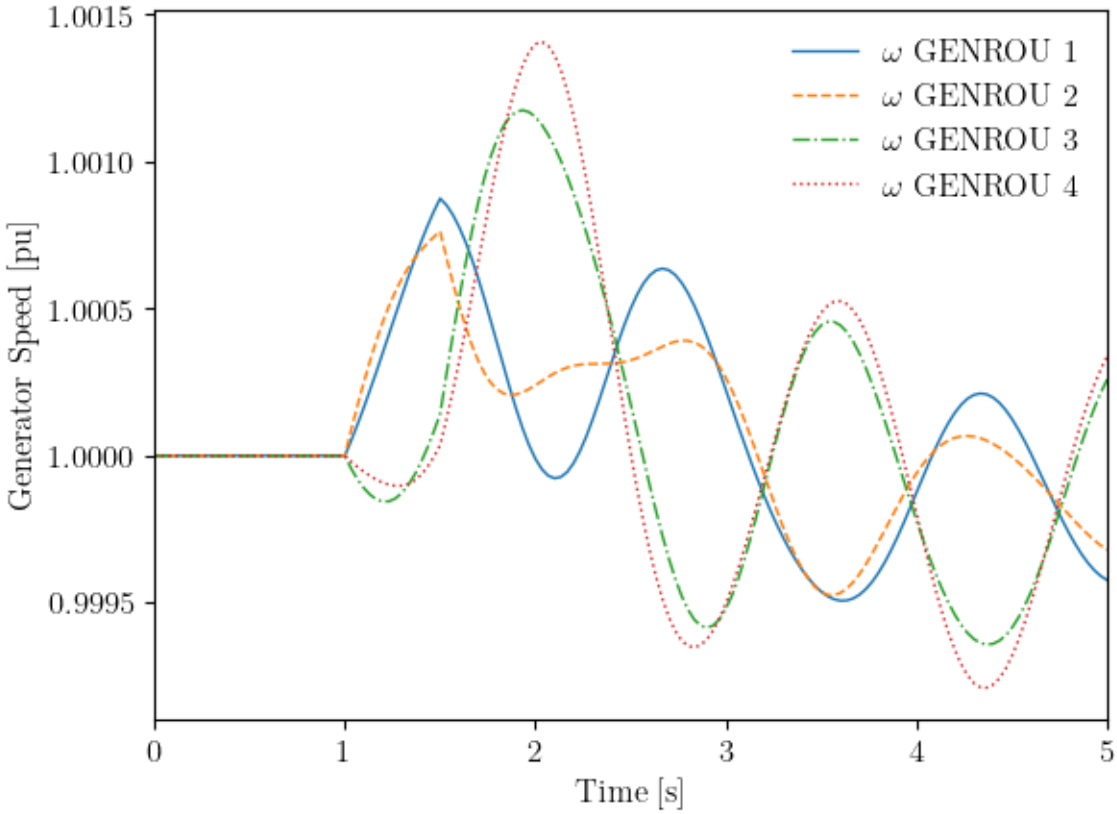
```
ss_toggle = andes.load(andes.get_case('kundur/kundur_full.xlsx'), setup=False)

# Disable existing Toggle
ss_toggle.Toggle.set('u', 1, 0)

# Trip line at t=1.0s, reclose at t=1.5s
ss_toggle.add('Toggle', model='Line', dev='Line_5', t=1.0)
ss_toggle.add('Toggle', model='Line', dev='Line_5', t=1.5)

ss_toggle.setup()
ss_toggle.PFlow.run()
ss_toggle.TDS.config.tf = 5
ss_toggle.TDS.run()
```

```
Working directory: "/home/docs/checkouts/readthedocs.org/user_builds/andes/
↪checkouts/stable/docs/source/tutorials"
> Reloaded generated Python code of module "pycode".
Parsing input file "/home/docs/checkouts/readthedocs.org/user_builds/andes/
↪envs/stable/lib/python3.11/site-packages/andes/cases/kundur/kundur_full.xlsx
↪"...
Input file parsed in 0.0298 seconds.
Connectivity check completed in 0.0001 seconds.
-> System connectivity check results:
  No islanded bus detected.
  System is interconnected.
  Each island has a slack bus correctly defined and enabled.
System internal structure set up in 0.0195 seconds.

-> Power flow calculation
           Numba: Off
   Sparse solver: KLU
 Solution method: NR method
Power flow initialized in 0.0030 seconds.
0: |F(x)| = 14.9282832
1: |F(x)| = 3.608627841
2: |F(x)| = 0.1701107882
3: |F(x)| = 0.002038626956
4: |F(x)| = 3.745103977e-07
Converged in 5 iterations in 0.0023 seconds.
Report saved to "kundur_full_out.txt" in 0.0006 seconds.
DAE compaction: removed 5 algebraic variable slots (m: 149 -> 144)
Initialization for dynamics completed in 0.0150 seconds.
Initialization was successful.
<Toggle Toggle_2>: Line.Line_5 status changed to 0 at t=1.0 sec.
<Toggle Toggle_3>: Line.Line_5 status changed to 1 at t=1.5 sec.
Simulation to t=5.00 sec completed in 0.2517 seconds.
Outputs to "kundur_full_out.lst" and "kundur_full_out.npz".
Outputs written in 0.0076 seconds.
```

```
  0%|          | 0/100 [00:00<?, ?%/s]
```

```
True
```

```
ss_toggle.TDS.plt.plot(ss_toggle.GENROU.omega, ylabel='Generator Speed [pu]')
```




```
(<Figure size 640x480 with 1 Axes>,
 <Axes: xlabel='Time [s]', ylabel='Generator Speed [pu]'>)
```

Toggle can operate on any model type. Common applications include:

- **Generator trip**: `ss.add('Toggle', model='GENROU', dev='GENROU_2', t=1.0)`
- **Load shedding**: `ss.add('Toggle', model='PQ', dev='PQ_1', t=2.0)`
- **Transformer trip**: `ss.add('Toggle', model='Line', dev='Line_10', t=0.5)`

Parameter Changes (Alter)

The `Alter` model changes parameter or service values at specified times during simulation. This is useful for studying the effect of setpoint changes, such as adjusting governor references or exciter voltage setpoints. The alteration is specified by the source field (`src`), the method (=, +, −, *, /), and the amount to apply.

```
ss_alter = andes.load(andes.get_case('kundur/kundur_full.xlsx'), setup=False)

# Disable existing Toggle
ss_alter.Toggle.set('u', 1, 0)

# Step change in governor setpoint at t=1.0s
ss_alter.add('Alter', model='TGOV1', dev=1, src='pref0',
             t=1.0, method='=', amount=0.8)
```

(continues on next page)

(continued from previous page)

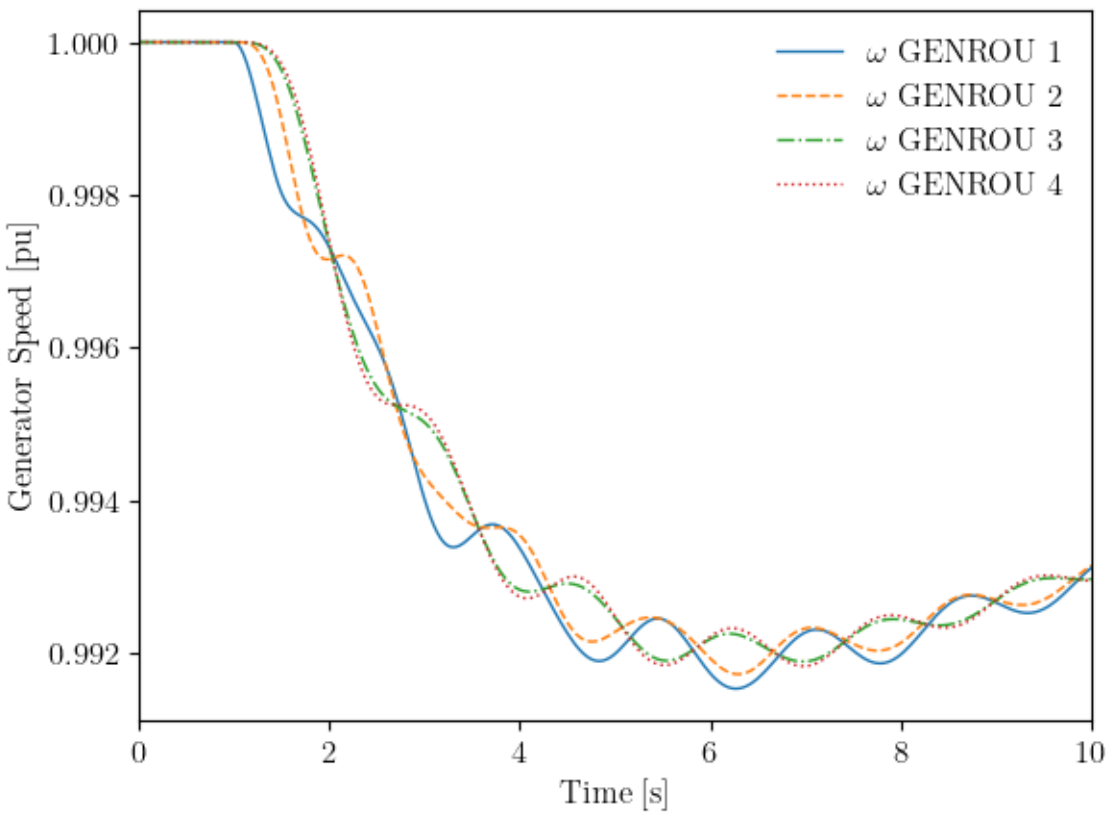
```
ss_alter.setup()
ss_alter.PFlow.run()
ss_alter.TDS.config.tf = 10
ss_alter.TDS.run()
```

```
Working directory: "/home/docs/checkouts/readthedocs.org/user_builds/andes/
↪checkouts/stable/docs/source/tutorials"
> Reloaded generated Python code of module "pycode".
Parsing input file "/home/docs/checkouts/readthedocs.org/user_builds/andes/
↪envs/stable/lib/python3.11/site-packages/andes/cases/kundur/kundur_full.xlsx
↪"...
Input file parsed in 0.0298 seconds.
Connectivity check completed in 0.0001 seconds.
-> System connectivity check results:
  No islanded bus detected.
  System is interconnected.
  Each island has a slack bus correctly defined and enabled.
System internal structure set up in 0.0194 seconds.

-> Power flow calculation
           Numba: Off
   Sparse solver: KLU
 Solution method: NR method
Power flow initialized in 0.0030 seconds.
0: |F(x)| = 14.9282832
1: |F(x)| = 3.608627841
2: |F(x)| = 0.1701107882
3: |F(x)| = 0.002038626956
4: |F(x)| = 3.745103977e-07
Converged in 5 iterations in 0.0023 seconds.
Report saved to "kundur_full_out.txt" in 0.0005 seconds.
DAE compaction: removed 5 algebraic variable slots (m: 149 -> 144)
Initialization for dynamics completed in 0.0152 seconds.
Initialization was successful.
<Alter Alter_2>: set TGOV1.1.pref0.v=0.8 at t=1. Previous value was 7.26803.
Simulation to t=10.00 sec completed in 0.6487 seconds.
Outputs to "kundur_full_out.lst" and "kundur_full_out.npz".
Outputs written in 0.0142 seconds.
```

```
  0%|          | 0/100 [00:00<?, ?%/s]
```

```
True
```

```
ss_alter.TDS.plt.plot(ss_alter.GENROU.omega, ylabel='Generator Speed [pu]')
```




```
(<Figure size 640x480 with 1 Axes>,
 <Axes: xlabel='Time [s]', ylabel='Generator Speed [pu]'>)
```

Altering Dynamic vs. Static Models

During TDS initialization, dynamic models disable their linked static models and take over power injection. For example, `GENROU` disables its linked `PV` generator, and `FLoad` disables its linked `PQ` load.

This means that altering a parameter on the disabled static model has no effect during simulation. A common mistake is trying to change generator output by altering `PV.p0`, which has no effect because `PV` has been disabled by `GENROU`. The correct approach is to alter the governor setpoint:

```
# Wrong: PV is disabled by GENROU, has no effect
ss.add('Alter', model='PV', dev=2, src='p0', t=1.0, method='+', amount=-0.5)

# Correct: alter the governor reference power
ss.add('Alter', model='TGOV1', dev=1, src='pref0', t=1.0, method='+', amount=-
↪0.5)
```

Tip: For interactive (non-timed) setpoint changes between simulation stages, the group-level setpoint API is recommended. It resolves the controller chain automatically so that the user does not need to know the specific governor or exciter model:

```
ss.SynGen.set_pref(ss, gen_idx, new_value)   # writes to governor pref0
ss.SynGen.set_vref(ss, gen_idx, new_value)    # writes to exciter vref0
ss.SynGen.set_paux(ss, gen_idx, new_value)    # writes to governor paux0
```

See *Dynamic Control and Setpoint Changes* for a complete example.

The table below lists common dynamic-static pairs and the correct target for alterations:

Dynamic Model	Disables	Wrong Target	Correct Target
GENROU/GENCLS	PV/Slack	PV.p0	TGOV1.pref0 or paux0
FLoad	PQ	PQ.Ppf	FLoad.pv0
ZIP	PQ	PQ.Ppf	ZIP.Sp

Note: When no dynamic model is attached, the static model remains active. For example, altering `PQ.Ppf` works for load shedding when no `FLoad` or `ZIP` is present. See *Frequency Response and Load Shedding* for an example.

2.5.4 Configuration Options

The TDS routine provides several configuration options that control simulation behavior. These can be accessed and modified through `ss.TDS.config`.

```
ss.TDS.config
```

```
OrderedDict([('linsolve', 0),
             ('method', 'trapezoid'),
             ('tol', 0.0001),
             ('t0', 0.0),
             ('tf', 20.0),
             ('fixt', 1),
             ('shrinkt', 1),
             ('honest', 0),
             ('tstep', 0.03333333333333333),
             ('max_iter', 15),
             ('refresh_event', 0),
             ('test_init', 1),
             ('check_conn', 1),
             ('criteria', 1),
             ('ddelta_limit', 180),
             ('g_scale', 1),
             ('reset_tiny', 1),
             ('qrt', 0),
             ('kqrt', 1.0),
             ('store_z', 0),
             ('store_f', 0),
```

(continues on next page)

(continued from previous page)

```
              ('store_h', 0),
              ('store_i', 0),
              ('limit_store', 0),
              ('max_store', 900),
              ('save_every', 1),
              ('save_mode', 'auto'),
              ('no_tqdm', 0),
              ('chatter_iter', 4),
              ('linesearch', 1),
              ('dtmax', 0),
              ('dtmin_adapt', 0),
              ('abstol', 1e-06),
              ('reltol', 0.001)])
```

The most commonly adjusted options are:

Option	Default	Description
`tf`	20	Simulation end time [seconds]
`tstep`	1/30	Integration time step [seconds]
`max_iter`	15	Maximum Newton iterations per step
`tol`	1e-6	Convergence tolerance
`fixt`	1	Use fixed (1) or variable (0) step size

For simulations with fast dynamics or discontinuities, you may need to reduce the time step to maintain accuracy and stability.

2.5.5 Accessing Results

After simulation, the `get_timeseries()` method on `ss.TDS` is the easiest way to extract results as a pandas DataFrame. Pass any variable object and it returns a DataFrame with time as the index and device indices as columns — no need to know whether the variable is a state or algebraic quantity.

Using `get_timeseries()`

```
# Generator speeds — returns a DataFrame automatically
omega_df = ss.TDS.get_timeseries(ss.GENROU.omega)
print(omega_df.head())
```

```
             1    2    3    4
0.000000  1.0  1.0  1.0  1.0
0.033333  1.0  1.0  1.0  1.0
0.066667  1.0  1.0  1.0  1.0
0.100000  1.0  1.0  1.0  1.0
0.133333  1.0  1.0  1.0  1.0
```

```
# Bus voltages — works the same way regardless of variable type
voltage_df = ss.TDS.get_timeseries(ss.Bus.v)
print(f"Voltage DataFrame shape: {voltage_df.shape}")
print(f"Columns (bus indices): {list(voltage_df.columns)}")
```

```
Voltage DataFrame shape: (303, 10)
Columns (bus indices): [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

The returned DataFrame can be used directly for analysis or export:

```
# Export to CSV
omega_df.to_csv('omega_results.csv')

# Quick statistics
print(omega_df.max())

# Slice a single device
omega_gen1 = omega_df[1]  # Generator with idx=1
```

Advanced: Direct DAE Access

Under the hood, time series data is stored in `ss.dae.ts`. State variables (like generator speed) are in `ss.dae.ts.x` and algebraic variables (like bus voltage) are in `ss.dae.ts.y`. Each variable has an `.a` attribute containing its address(es) into these arrays.

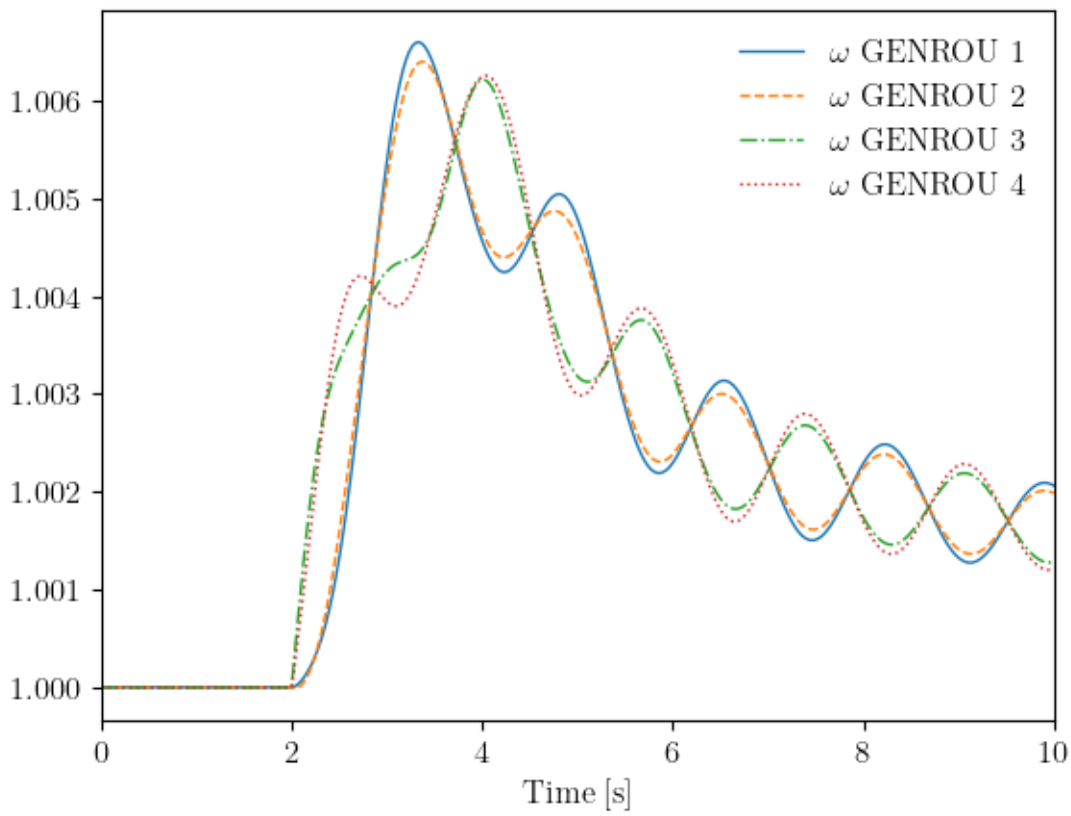
```
# State variable: use dae.ts.x
omega_addr = ss.GENROU.omega.a        # array([4, 5, 6, 7])
omega_ts = ss.dae.ts.x[:, omega_addr] # shape: (n_steps, 4)

# Algebraic variable: use dae.ts.y
v_addr = ss.Bus.v.a
voltage_ts = ss.dae.ts.y[:, v_addr]   # shape: (n_steps, 10)

# Time vector
t = ss.dae.ts.t
```

This direct access is useful when you need maximum performance or are working with custom numerical analysis.

2.5.6 Plotting Results

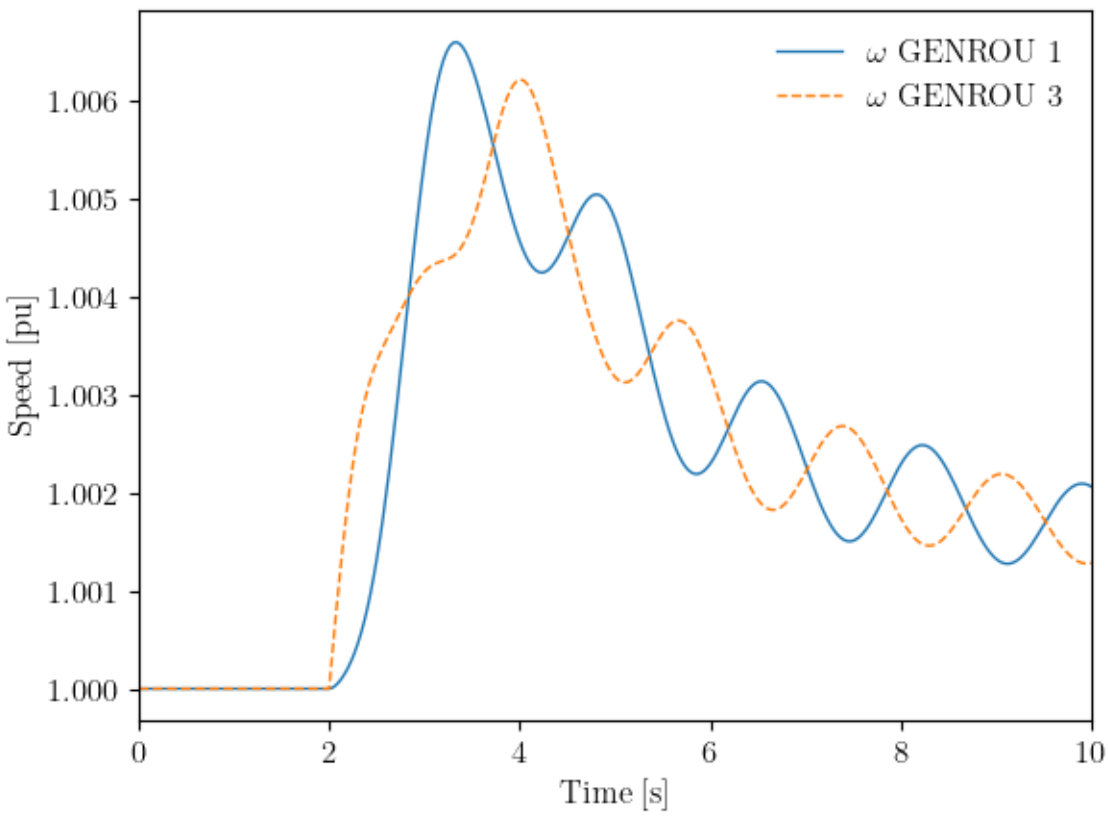
ANDES provides a built-in plotter accessible through `ss.TDS.plt` that simplifies visualization of simulation results. The `plot()` method accepts variable objects and automatically handles address lookup and data extraction.

```
# Plot all generator speeds
ss.TDS.plt.plot(ss.GENROU.omega)
```




```
(<Figure size 640x480 with 1 Axes>, <Axes: xlabel='Time [s]'>)
```

```
# Plot specific generators using address indices
ss.TDS.plt.plot(ss.GENROU.omega, a=[0, 2], ylabel='Speed [pu]')
```




```
(<Figure size 640x480 with 1 Axes>,
 <Axes: xlabel='Time [s]', ylabel='Speed [pu]'>)
```

Exporting Results

Results can be exported to CSV format for further analysis in other tools.

```
ss.TDS.plt.export_csv('kundur_results.csv')
```

```
CSV data saved to "kundur_results.csv".
```

```
'kundur_results.csv'
```

2.5.7 Multi-Stage Simulation

For complex scenarios, the simulation can be run in multiple stages with setpoint changes between stages. After completing one stage, the end time is updated and `run()` is called again. ANDES continues from where it left off, preserving all state variables.

The group-level setpoint API is used to apply changes between stages. See *Dynamic Control and Setpoint Changes* for a detailed tutorial.

```
ss_multi = andes.load(andes.get_case('kundur/kundur_full.xlsx'), setup=False)
ss_multi.Toggle.set('u', 1, 0)  # Disable existing toggle
ss_multi.setup()
ss_multi.PFlow.run()

# Stage 1: Run to t=2s
ss_multi.TDS.config.tf = 2.0
ss_multi.TDS.run()

# Apply setpoint change between stages using the group-level API
gen_idx = ss_multi.GENROU.idx.v[0]
ss_multi.SynGen.set_paux(ss_multi, gen_idx, 0.05)

# Stage 2: Continue to t=10s
ss_multi.TDS.config.tf = 10.0
ss_multi.TDS.run()
```

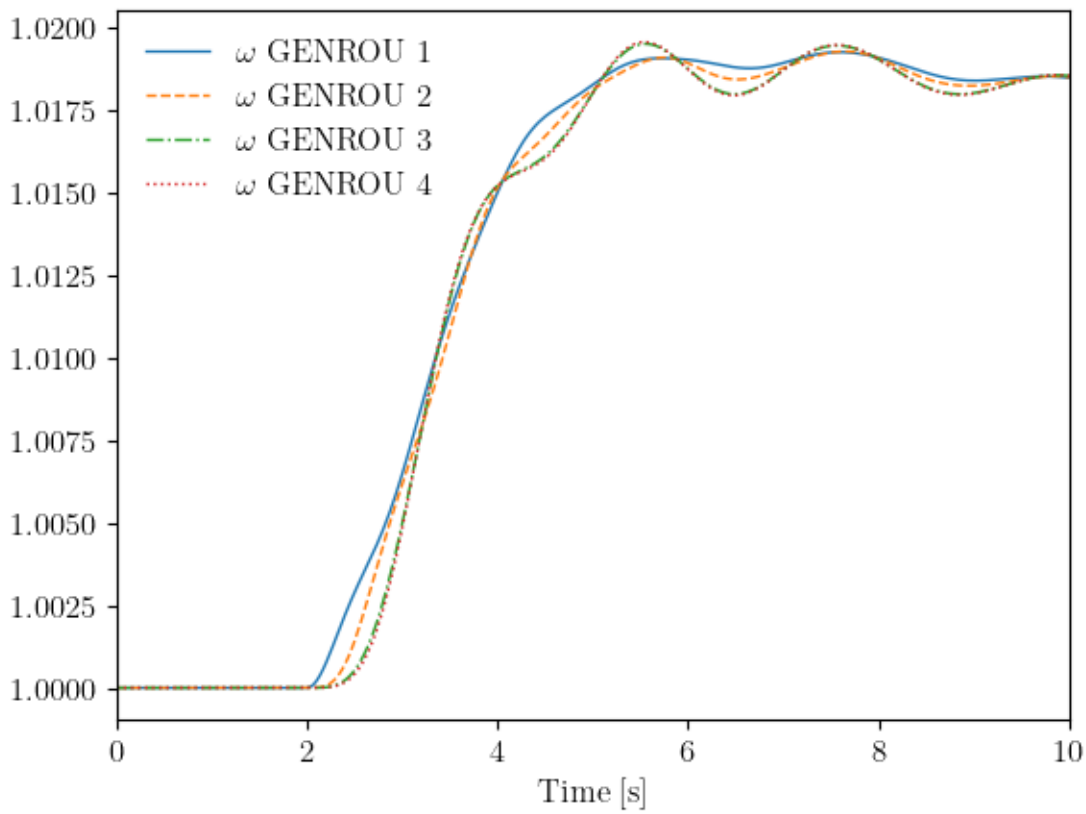
```
Working directory: "/home/docs/checkouts/readthedocs.org/user_builds/andes/
↪checkouts/stable/docs/source/tutorials"
> Reloaded generated Python code of module "pycode".
Parsing input file "/home/docs/checkouts/readthedocs.org/user_builds/andes/
↪envs/stable/lib/python3.11/site-packages/andes/cases/kundur/kundur_full.xlsx
↪"...
Input file parsed in 0.0294 seconds.
Connectivity check completed in 0.0001 seconds.
-> System connectivity check results:
  No islanded bus detected.
  System is interconnected.
  Each island has a slack bus correctly defined and enabled.
System internal structure set up in 0.0190 seconds.

-> Power flow calculation
           Numba: Off
   Sparse solver: KLU
 Solution method: NR method
Power flow initialized in 0.0029 seconds.
0: |F(x)| = 14.9282832
1: |F(x)| = 3.608627841
2: |F(x)| = 0.1701107882
3: |F(x)| = 0.002038626956
4: |F(x)| = 3.745103977e-07
Converged in 5 iterations in 0.0023 seconds.
Report saved to "kundur_full_out.txt" in 0.0006 seconds.
DAE compaction: removed 5 algebraic variable slots (m: 149 -> 144)
Initialization for dynamics completed in 0.0148 seconds.
Initialization was successful.
Simulation to t=2.00 sec completed in 0.0575 seconds.
Outputs to "kundur_full_out.lst" and "kundur_full_out.npz".
Outputs written in 0.0037 seconds.
Simulation to t=10.00 sec completed in 0.4868 seconds.
Outputs to "kundur_full_out.lst" and "kundur_full_out.npz".
Outputs written in 0.0140 seconds.
```

```
  0%|          | 0/100 [00:00<?, ?%/s]
```

```
  0%|          | 0/100 [00:00<?, ?%/s]
```

```
True
```

```
# Plot shows both stages seamlessly
ss_multi.TDS.plt.plot(ss_multi.GENROU.omega)
```




```
(<Figure size 640x480 with 1 Axes>, <Axes: xlabel='Time [s]'>)
```

2.5.8 Output Files

TDS automatically generates output files containing simulation results:

File	Description
`*_out.lst`	Variable listing with indices and names
`*_out.npz`	Compressed NumPy archive with time series data

These files can be loaded later for post-processing without re-running the simulation.

```
import numpy as np

# Load saved results
data = np.load(ss.files.npz)
print("Available arrays:", list(data.keys()))
```

```
Available arrays: ['data']
```

2.5.9 Troubleshooting

Simulation Fails to Converge

If the simulation fails during integration, the Newton iteration at a particular time step did not converge within the allowed iterations. This usually indicates:

1. **Step size too large**: Reduce `tstep` to capture faster dynamics
2. **Severe disturbance**: The system may be transiently unstable
3. **Model limitations**: Some operating points may exceed model validity ranges

```
# Typical remedies for convergence issues
ss.TDS.config.tstep = 0.001  # Smaller step size
ss.TDS.config.max_iter = 30  # More iterations allowed
```

Checking Initialization

Before running a full simulation, you can initialize TDS without running to check for any initialization errors. This helps identify problems with initial conditions before committing to a long simulation.

```
ss_check = andes.load(andes.get_case('kundur/kundur_full.xlsx'))
ss_check.PFlow.run()
ss_check.TDS.init()  # Initialize only, don't run
```

```
Working directory: "/home/docs/checkouts/readthedocs.org/user_builds/andes/
↪checkouts/stable/docs/source/tutorials"
> Reloaded generated Python code of module "pycode".
Parsing input file "/home/docs/checkouts/readthedocs.org/user_builds/andes/
↪envs/stable/lib/python3.11/site-packages/andes/cases/kundur/kundur_full.xlsx
↪"...
Input file parsed in 0.0304 seconds.
Connectivity check completed in 0.0001 seconds.
-> System connectivity check results:
  No islanded bus detected.
  System is interconnected.
  Each island has a slack bus correctly defined and enabled.
System internal structure set up in 0.0199 seconds.
```

(continues on next page)

(continued from previous page)

```
-> Power flow calculation
           Numba: Off
   Sparse solver: KLU
 Solution method: NR method
Power flow initialized in 0.0030 seconds.
0: |F(x)| = 14.9282832
1: |F(x)| = 3.608627841
2: |F(x)| = 0.1701107882
3: |F(x)| = 0.002038626956
4: |F(x)| = 3.745103977e-07
Converged in 5 iterations in 0.0024 seconds.
Report saved to "kundur_full_out.txt" in 0.0006 seconds.
DAE compaction: removed 5 algebraic variable slots (m: 149 -> 144)
Initialization for dynamics completed in 0.0152 seconds.
Initialization was successful.
```

```
array([ 1.41994833,  1.12395571,  0.93892059,  1.21137566,  1.        ,
        1.        ,  1.        ,  1.        ,  0.86626507,  0.94860464,
        0.95125532,  0.868703  ,  0.5080822 ,  0.45912205,  0.45770681,
        0.50085419,  0.70142377,  0.77725184,  0.77932427,  0.71147986,
        0.72456939,  0.65474796,  0.65272971,  0.71426163,  7.26802908,
        7.        ,  7.        ,  7.        ,  7.26802908,  7.        ,
        7.        ,  7.        ,  1.89652317,  2.0195597 ,  2.02582439,
        1.85134764,  1.        ,  1.        ,  1.        ,  1.        ,
        0.09482616,  0.10097799,  0.10129122,  0.09256738,  1.89652317,
        2.0195597 ,  2.02582439,  1.85134764,  1.89652317,  2.0195597 ,
        2.02582439,  1.85134764,  0.57025492,  0.37796171,  0.1957715 ,
        0.3777206 ,  0.4825648 ,  0.293535  ,  0.1425481 , -0.03712551,
        0.1113441 ,  0.29331307,  1.        ,  1.        ,  1.        ,
        1.        ,  0.98337472,  0.96908586,  0.95621812,  0.95400019,
        0.96856366,  0.98377143,  6.18154863,  6.42573035,  6.44741438,
        5.89586785,  3.97629545,  3.59312906,  3.58205329,  3.91972845,
        0.75107803,  0.67870216,  0.67661007,  0.74039315,  0.66021345,
        0.73441363,  0.73634151,  0.67217407,  7.26802908,  7.        ,
        7.        ,  7.        ,  7.26802908,  7.        ,  7.        ,
        7.        ,  1.89652317,  2.0195597 ,  2.02582439,  1.85134764,
        1.89652317,  2.0195597 ,  2.02582439,  1.85134764,  7.26802908,
        7.        ,  7.        ,  7.        ,  1.09463337,  2.28047954,
        2.32384553,  1.0609094 ,  0.66021345,  0.73441363,  0.73634151,
        0.67217407, -0.75107803, -0.67870216, -0.67661007, -0.74039315,
        0.64062538,  0.57889301,  0.57710859,  0.63151181,  0.83192313,
        0.91290614,  0.91543635,  0.83594818,  1.04999856,  1.08097861,
        1.08216359,  1.04767195,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,  7.26802908,
        7.        ,  7.        ,  7.        ,  1.        ,  1.        ,
        1.        ,  1.        ,  0.04037794,  0.03888889,  0.03888889,
        0.03888889,  0.        ,  0.        ,  0.        ,  0.        ,
        7.26802908,  7.        ,  7.        ,  7.        ,  7.26802908,
        7.        ,  7.        ,  7.        ,  1.        ,  1.        ,
        1.        ,  1.        ,  1.89652317,  2.0195597 ,  2.02582439,
```

(continues on next page)

(continued from previous page)

```
       1.85134764,  1.09482616,  1.10097799,  1.10129122,  1.09256738,
       0.        ,  0.        ,  0.        ,  0.        ,  0.09482616,
       0.10097799,  0.10129122,  0.09256738,  0.09482616,  0.10097799,
       0.10129122,  0.09256738,  0.        ,  0.        ,  0.        ,
       0.        ])
```

Flat Run Test

A "flat run" without any disturbances verifies that the system stays at equilibrium. If variables drift during a flat run, there may be inconsistencies in the initial conditions or model parameters.

```
andes run case.xlsx -r tds --flat
```

2.5.10 Cleanup

```
!andes misc -C
!rm -f kundur_results.csv
```

```
"/home/docs/checkouts/readthedocs.org/user_builds/andes/checkouts/stable/docs/
↪source/tutorials/kundur_full_out.txt" removed.
"/home/docs/checkouts/readthedocs.org/user_builds/andes/checkouts/stable/docs/
↪source/tutorials/kundur_full_out.lst" removed.
"/home/docs/checkouts/readthedocs.org/user_builds/andes/checkouts/stable/docs/
↪source/tutorials/kundur_full_out.npz" removed.
```

2.5.11 Next Steps

- *Data and File Formats* - Loading different file formats and modifying parameters
- *Plotting Results* - Advanced plotting and result visualization
- *Eigenvalue Analysis* - Small-signal stability analysis

2.6 Data and File Formats

ANDES supports multiple file formats for loading power system data, including its native Excel format, PSS/E files, and MATPOWER cases. This tutorial covers loading data from different sources, inspecting and modifying parameters, and saving modified cases.

Understanding how to work with data is essential for practical power system studies, where you often need to adapt standard test cases to specific scenarios or import data from other simulation tools.

2.6.1 Setup

```
%matplotlib inline

import andes
import pandas as pd

andes.config_logger(stream_level=20)

# Configure pandas to show all columns
pd.options.display.max_columns = 20
```

2.6.2 Loading ANDES Excel Files

The native ANDES Excel format (`.xlsx`) is the best-supported format and is recommended for most use cases. Each sheet in the workbook corresponds to a model type (Bus, Line, GENROU, etc.), and each row represents a device instance.

Use `andes.load()` to create a System object from a case file. The `andes.get_case()` function returns the path to built-in test cases.

```
ss = andes.load(andes.get_case('kundur/kundur_full.xlsx'))
ss
```

```
Working directory: "/home/docs/checkouts/readthedocs.org/user_builds/andes/
↪checkouts/stable/docs/source/tutorials"
> Loaded generated Python code in "/home/docs/.andes/pycode".
Parsing input file "/home/docs/checkouts/readthedocs.org/user_builds/andes/
↪envs/stable/lib/python3.11/site-packages/andes/cases/kundur/kundur_full.xlsx
↪"...
Input file parsed in 0.0313 seconds.
Connectivity check completed in 0.0001 seconds.
-> System connectivity check results:
  No islanded bus detected.
  System is interconnected.
  Each island has a slack bus correctly defined and enabled.
System internal structure set up in 0.0203 seconds.
```

```
<andes.system.facade.System at 0x73deecee8990>
```

The System object provides a summary of all loaded models. Each model can be accessed as an attribute of the system, for example `ss.Bus`, `ss.Line`, or `ss.GENROU`.

2.6.3 Loading PSS/E Files

PSS/E is the industry-standard power system simulation software from Siemens. ANDES can import PSS/E data files directly, supporting RAW files for power flow data and DYR files for dynamic model parameters.

File Type	Extension	Contents
RAW	`.raw`	Power flow data (buses, generators, loads, branches)
DYR	`.dyr`	Dynamic model parameters (exciters, governors, stabilizers)

To load PSS/E files, pass the RAW file path to `andes.load()` and use the `addfile` argument for the DYR file. The `addfile` argument accepts a single file path or a list of paths.

```
raw_path = andes.get_case('kundur/kundur.raw')
dyr_path = andes.get_case('kundur/kundur_full.dyr')

ss_psse = andes.load(raw_path, addfile=dyr_path)
ss_psse.PFlow.run()
```

```
Working directory: "/home/docs/checkouts/readthedocs.org/user_builds/andes/
↪checkouts/stable/docs/source/tutorials"
> Reloaded generated Python code of module "pycode".
Parsing input file "/home/docs/checkouts/readthedocs.org/user_builds/andes/
↪envs/stable/lib/python3.11/site-packages/andes/cases/kundur/kundur.raw"...
  MODIFIED KUNDUR'S TWO-AREA TEST SYSTEM, DISTRIBUTED WITH ANDES
  SEE THE BOOK "POWER SYSTEM STABILITY AND CONTROL" FOR ORIGINAL DATA
Input file parsed in 0.0542 seconds.
Parsing additional file "/home/docs/checkouts/readthedocs.org/user_builds/
↪andes/envs/stable/lib/python3.11/site-packages/andes/cases/kundur/kundur_
↪full.dyr"...
Addfile parsed in 0.0720 seconds.
Connectivity check completed in 0.0001 seconds.
-> System connectivity check results:
  No islanded bus detected.
  System is interconnected.
  Each island has a slack bus correctly defined and enabled.
System internal structure set up in 0.0186 seconds.

-> Power flow calculation
           Numba: Off
   Sparse solver: KLU
 Solution method: NR method
Power flow initialized in 0.0036 seconds.
0: |F(x)| = 3.175850023
1: |F(x)| = 3.176155228e-08
Converged in 2 iterations in 0.0010 seconds.
Report saved to "kundur_out.txt" in 0.0006 seconds.
```

```
True
```

Command Line Usage

PSS/E files can also be loaded from the command line. The `-a` flag can be repeated to load multiple additional files:

```
# Power flow only (RAW file)
andes run kundur.raw

# With dynamic models (RAW + DYR)
andes run kundur.raw --addfile kundur.dyr -r tds
```

Cross-Format Loading

The `addfile` argument is not limited to PSS/E DYR files. Any supported format (xlsx, json, matpower) can be used as an additional file. For example, an ANDES Excel file containing only dynamic models can be loaded on top of a RAW power flow case:

```
ss = andes.load('case.raw', addfile='dynamics.xlsx')
```

Converting to ANDES Format

PSS/E files can be converted to the native ANDES Excel format for easier editing:

```
andes run kundur.raw --addfile kundur.dyr --convert xlsx
```

2.6.4 Inspecting Parameters

After loading a system, you can inspect model parameters in several ways. Each model provides an `as_df()` method that returns a pandas DataFrame containing all parameter values.

Viewing Model Parameters

```
# View GENROU parameters as DataFrame
ss.GENROU.as_df()
```

```
     idx    u      name  bus  gen   coi  coi2     Sn    Vn    fn  ...  gammaq␣
↪ \
uid                                                               ...
0      1  1.0  GENROU_1    1    1  None  None  900.0  20.0  60.0  ...     1.0
1      2  1.0  GENROU_2    2    2  None  None  900.0  20.0  60.0  ...     1.0
2      3  1.0  GENROU_3    3    3  None  None  900.0  20.0  60.0  ...     1.0
3      4  1.0  GENROU_4    4    4  None  None  900.0  20.0  60.0  ...     1.0

      xd        xq       xd2       xq1       xq2  Td10  Td20  Tq10  Tq20
uid
```

(continues on next page)

(continued from previous page)

```
0    0.2  0.188889  0.027778  0.061111  0.027778   8.0  0.03   0.4  0.05
1    0.2  0.188889  0.027778  0.061111  0.027778   8.0  0.03   0.4  0.05
2    0.2  0.188889  0.027778  0.061111  0.027778   8.0  0.03   0.4  0.05
3    0.2  0.188889  0.027778  0.061111  0.027778   8.0  0.03   0.4  0.05

[4 rows x 30 columns]
```

The DataFrame shows all parameters for each device, with the unique device index (`idx`) in the first column. Parameters are shown after conversion to per-unit on the system base.

To see the original input values before per-unit conversion, use `as_df(vin=True)`.

```
# Original input values
ss.GENROU.as_df(vin=True)[['idx', 'Sn', 'M', 'xd', 'xq']]
```

```
     idx     Sn      M   xd   xq
uid
0      1  900.0  13.00  1.8  1.7
1      2  900.0  13.00  1.8  1.7
2      3  900.0  12.35  1.8  1.7
3      4  900.0  12.35  1.8  1.7
```

Getting Specific Parameters

To retrieve a specific parameter value for a device, use the `get()` method. This is useful when you need to read a value for use in calculations or comparisons.

```
# Get the inertia constant M for generator 1
M_value = ss.GENROU.get('M', 1)
print(f"Inertia M for GENROU 1: {M_value} pu")
```

```
Inertia M for GENROU 1: 117.0 pu
```

Accessing Parameter Arrays

Each parameter is stored as a `NumParam` object with a `v` attribute containing the numerical values as a NumPy array. This provides direct access for bulk operations.

```
# Get all M values as an array
print("All generator inertia constants:")
print(ss.GENROU.M.v)
```

```
All generator inertia constants:
[117.   117.   111.15 111.15]
```

2.6.5 Modifying Parameters

Parameters can be modified using the `set()` method. By default, `set()` writes values in **system base** — the same base returned by `get()`. This means the read-modify-write cycle is straightforward:

```
h = ss.GENROU.get('M', 1)          # read system-base value
ss.GENROU.set('M', 1, h * 1.1)     # write back in system base
```

To provide values in **device/input base** (as they appear in the case file), use `base='device'`. This converts through `pu_coeff` and also updates the `vin` record so that dumped case files remain consistent.

Using set() with device base

```
# Load a fresh system
ss_mod = andes.load(andes.get_case('kundur/kundur_full.xlsx'))

# Check original value
print(f"Original M: {ss_mod.GENROU.get('M', 1, attr='vin')}")

# Modify the inertia constant (value is in device/input base, like the case␣
↪file)
ss_mod.GENROU.set('M', 1, 10.0, base='device')

# Verify the change
print(f"Modified M: {ss_mod.GENROU.get('M', 1, attr='vin')}")
```

```
Original M: 13.0
Modified M: 10.0
```

```
Working directory: "/home/docs/checkouts/readthedocs.org/user_builds/andes/
↪checkouts/stable/docs/source/tutorials"
> Reloaded generated Python code of module "pycode".
Parsing input file "/home/docs/checkouts/readthedocs.org/user_builds/andes/
↪envs/stable/lib/python3.11/site-packages/andes/cases/kundur/kundur_full.xlsx
↪"...
Input file parsed in 0.0307 seconds.
Connectivity check completed in 0.0001 seconds.
-> System connectivity check results:
  No islanded bus detected.
  System is interconnected.
  Each island has a slack bus correctly defined and enabled.
System internal structure set up in 0.0195 seconds.
```

When `base='device'` is used, the value should be in the same units as the input file (before per-unit conversion). ANDES will automatically convert it to the appropriate system base and update the stored input value (`vin`).

When `base` is omitted (the default), the value is written directly in system base — matching what `get()` returns.

Direct Array Modification

For bulk modifications or performance-critical operations, you can directly modify the parameter arrays. When doing so, always use in-place assignment with slicing to avoid memory issues.

```
# Scale all PQ loads by 10%
ss_mod.PQ.Ppf.v[:] *= 1.1
ss_mod.PQ.Qpf.v[:] *= 1.1

print("Scaled PQ loads:")
print(ss_mod.PQ.Ppf.v)
```

```
Scaled PQ loads:
[0. 0.]
```

Verifying Modified Parameters

After modifying parameters, call `as_df(vin=True)` to inspect the updated input values.

```
ss_mod.GENROU.as_df(vin=True)[['idx', 'M']]
```

```
     idx      M
uid
0      1  10.00
1      2  13.00
2      3  12.35
3      4  12.35
```

2.6.6 Modifying Device Status

Each device has a connectivity status parameter `u` that can be set to 0 (disabled) or 1 (enabled). This is useful for studying contingencies without removing devices from the data file.

To disable a device before running simulation, load with `setup=False`, modify the status, then call `setup()`.

```
ss_cont = andes.load(andes.get_case('kundur/kundur_full.xlsx'), setup=False)

# Disable a transmission line
ss_cont.Line.set('u', 'Line_3', 0)

# Complete system setup and run power flow
ss_cont.setup()
ss_cont.PFlow.run()
```

```
Working directory: "/home/docs/checkouts/readthedocs.org/user_builds/andes/
↪checkouts/stable/docs/source/tutorials"
> Reloaded generated Python code of module "pycode".
```

(continues on next page)

(continued from previous page)

```
Parsing input file "/home/docs/checkouts/readthedocs.org/user_builds/andes/
↪envs/stable/lib/python3.11/site-packages/andes/cases/kundur/kundur_full.xlsx
↪"...
Input file parsed in 0.0308 seconds.
Connectivity check completed in 0.0002 seconds.
-> System connectivity check results:
  No islanded bus detected.
  System is interconnected.
  Each island has a slack bus correctly defined and enabled.
System internal structure set up in 0.0194 seconds.

-> Power flow calculation
           Numba: Off
   Sparse solver: KLU
 Solution method: NR method
Power flow initialized in 0.0031 seconds.
0: |F(x)| = 14.9282832
1: |F(x)| = 3.579044433
2: |F(x)| = 0.119268955
3: |F(x)| = 0.03278820195
4: |F(x)| = 2.880943096e-05
5: |F(x)| = 3.93747257e-11
Converged in 6 iterations in 0.0029 seconds.
Report saved to "kundur_full_out.txt" in 0.0005 seconds.
```

```
True
```

2.6.7 Saving Modified Cases

After modifying parameters, you can save the system to a new file. This is useful for creating variations of a base case for parameter studies.

```
# Save to new Excel file
andes.io.xlsx.write(ss_mod, 'modified_case.xlsx', overwrite=True)
```

```
xlsx file written to "modified_case.xlsx"
```

```
True
```

2.6.8 Accessing Variables After Simulation

After running power flow or time-domain simulation, solved variable values can be accessed through the model objects. Each variable has a `v` attribute containing its current value.

Power Flow Results

```
ss.PFlow.run()

# Bus voltage magnitudes
print("Bus voltages (pu):")
print(ss.Bus.v.v)
```

```
Bus voltages (pu):
[1.         1.         1.         1.         0.98337472 0.96908586
 0.95621812 0.95400019 0.96856366 0.98377143]
```

```
-> Power flow calculation
           Numba: Off
   Sparse solver: KLU
 Solution method: NR method
Power flow initialized in 0.0036 seconds.
0: |F(x)| = 14.9282832
1: |F(x)| = 3.608627841
2: |F(x)| = 0.1701107882
3: |F(x)| = 0.002038626956
4: |F(x)| = 3.745103977e-07
Converged in 5 iterations in 0.0024 seconds.
Report saved to "kundur_full_out.txt" in 0.0005 seconds.
```

Variable Types

Variables are classified as either state (differential) or algebraic:

- **State variables** (stored in `ss.dae.x`): Generator angles, speeds, flux linkages
- **Algebraic variables** (stored in `ss.dae.y`): Bus voltages, angles, power injections

Each variable object has an `a` attribute containing its address(es) in the system arrays.

```
# Generator speed is a state variable
print(f"GENROU.omega type: {type(ss.GENROU.omega).__name__}")
print(f"GENROU.omega addresses: {ss.GENROU.omega.a}")

# Bus voltage is an algebraic variable
print(f"\nBus.v type: {type(ss.Bus.v).__name__}")
print(f"Bus.v addresses: {ss.Bus.v.a}")
```

```
GENROU.omega type: State
GENROU.omega addresses: []

Bus.v type: Algeb
Bus.v addresses: [10 11 12 13 14 15 16 17 18 19]
```

2.6.9 Working with Device Indices

Each device is identified by a unique index (`idx`). The index can be any hashable type (string, integer, etc.) and is used consistently across ANDES for referencing devices.

Listing Device Indices

```
# Get all Line indices
print("Line indices:")
print(ss.Line.idx.v)
```

```
Line indices:
['Line_0', 'Line_1', 'Line_2', 'Line_3', 'Line_4', 'Line_5', 'Line_6', 'Line_7
↪', 'Line_8', 'Line_9', 'Line_10', 'Line_11', 'Line_12', 'Line_13', 'Line_14
↪']
```

```
# Get all GENROU indices
print("GENROU indices:")
print(ss.GENROU.idx.v)
```

```
GENROU indices:
[1, 2, 3, 4]
```

Converting Index to Internal UID

Internally, ANDES uses zero-based integer UIDs for array indexing. The `idx2uid()` method converts an external index to its internal UID.

```
# Convert idx to uid
uid = ss.GENROU.idx2uid(1)
print(f"GENROU idx=1 has uid={uid}")

# Access parameter by uid
print(f"M value at uid {uid}: {ss.GENROU.M.v[uid]}")
```

```
GENROU idx=1 has uid=0
M value at uid 0: 117.0
```

2.6.10 Parameter Units

Parameters are typically specified in per-unit on the device or system base. Common conventions are:

Parameter Type	Typical Unit
Power (`p0`, `q0`)	pu on system MVA base
Voltage (`v0`)	pu on bus nominal voltage
Impedance (`R`, `X`)	pu on system base
Inertia (`H`, `M`)	seconds (M = 2H)
Time constants	seconds

Consult the model documentation for specific parameter definitions and units.

2.6.11 Cleanup

```
!andes misc -C
!rm -f modified_case.xlsx
```

```
"/home/docs/checkouts/readthedocs.org/user_builds/andes/checkouts/stable/docs/
↪source/tutorials/kundur_full_out.txt" removed.
"/home/docs/checkouts/readthedocs.org/user_builds/andes/checkouts/stable/docs/
↪source/tutorials/kundur_out.txt" removed.
```

2.6.12 Next Steps

- *Plotting Results* - Visualizing simulation results
- *Eigenvalue Analysis* - Small-signal stability analysis
- *Parameter Sweeps and Batch Processing* - Running batch simulations

2.7 Plotting Results

Effective visualization is essential for understanding power system dynamics. After running a time-domain simulation, you typically want to examine how variables like generator speeds, bus voltages, and power flows change over time. ANDES provides a built-in plotting interface through `ss.TDS.plt` that makes it easy to create publication-quality figures with minimal code.

This tutorial covers the plotting capabilities available in ANDES, from basic variable plotting to customization options and exporting results. By the end, you will be able to create informative plots that clearly communicate simulation results.

2.7.1 Setup and Run Simulation

Before we can plot results, we need simulation data. Let us run a time-domain simulation on the Kundur two-area system, which contains a line trip disturbance that triggers interesting oscillatory dynamics.

```
%matplotlib inline

import andes

andes.config_logger(stream_level=20)

# Load system and run power flow
ss = andes.load(andes.get_case('kundur/kundur_full.xlsx'))
ss.PFlow.run()

# Run time-domain simulation
ss.TDS.config.tf = 10
ss.TDS.run()
```

```
Working directory: "/home/docs/checkouts/readthedocs.org/user_builds/andes/
↪checkouts/stable/docs/source/tutorials"
> Loaded generated Python code in "/home/docs/.andes/pycode".
Parsing input file "/home/docs/checkouts/readthedocs.org/user_builds/andes/
↪envs/stable/lib/python3.11/site-packages/andes/cases/kundur/kundur_full.xlsx
↪"...
Input file parsed in 0.0315 seconds.
Connectivity check completed in 0.0001 seconds.
-> System connectivity check results:
  No islanded bus detected.
  System is interconnected.
  Each island has a slack bus correctly defined and enabled.
System internal structure set up in 0.0205 seconds.

-> Power flow calculation
           Numba: Off
   Sparse solver: KLU
 Solution method: NR method
Power flow initialized in 0.0031 seconds.
0: |F(x)| = 14.9282832
1: |F(x)| = 3.608627841
2: |F(x)| = 0.1701107882
3: |F(x)| = 0.002038626956
4: |F(x)| = 3.745103977e-07
Converged in 5 iterations in 0.0025 seconds.
Report saved to "kundur_full_out.txt" in 0.0008 seconds.
DAE compaction: removed 5 algebraic variable slots (m: 149 -> 144)
Initialization for dynamics completed in 0.0152 seconds.
Initialization was successful.
<Toggle 1>: Line.Line_8 status changed to 0 at t=2.0 sec.
Simulation to t=10.00 sec completed in 0.5439 seconds.
Outputs to "kundur_full_out.lst" and "kundur_full_out.npz".
Outputs written in 0.0142 seconds.
```

```
  0%|          | 0/100 [00:00<?, ?%/s]
```

```
True
```

2.7.2 The Plotter Object

When running ANDES interactively (in Jupyter Notebook or IPython), a plotter object is automatically created at `ss.TDS.plt` after time-domain simulation completes. This object provides methods for plotting variables, finding variable indices, and exporting data.

You can verify that the plotter was created successfully by checking its type.

```
ss.TDS.plt
```

```
<andes.plot.TDSData at 0x7d3c2fc56b90>
```

If `ss.TDS.plt` is `None` (which can happen if the interactive environment detection fails), you can manually create it by calling `ss.TDS.load_plotter()`. This step is normally not needed but is useful to know about for troubleshooting.

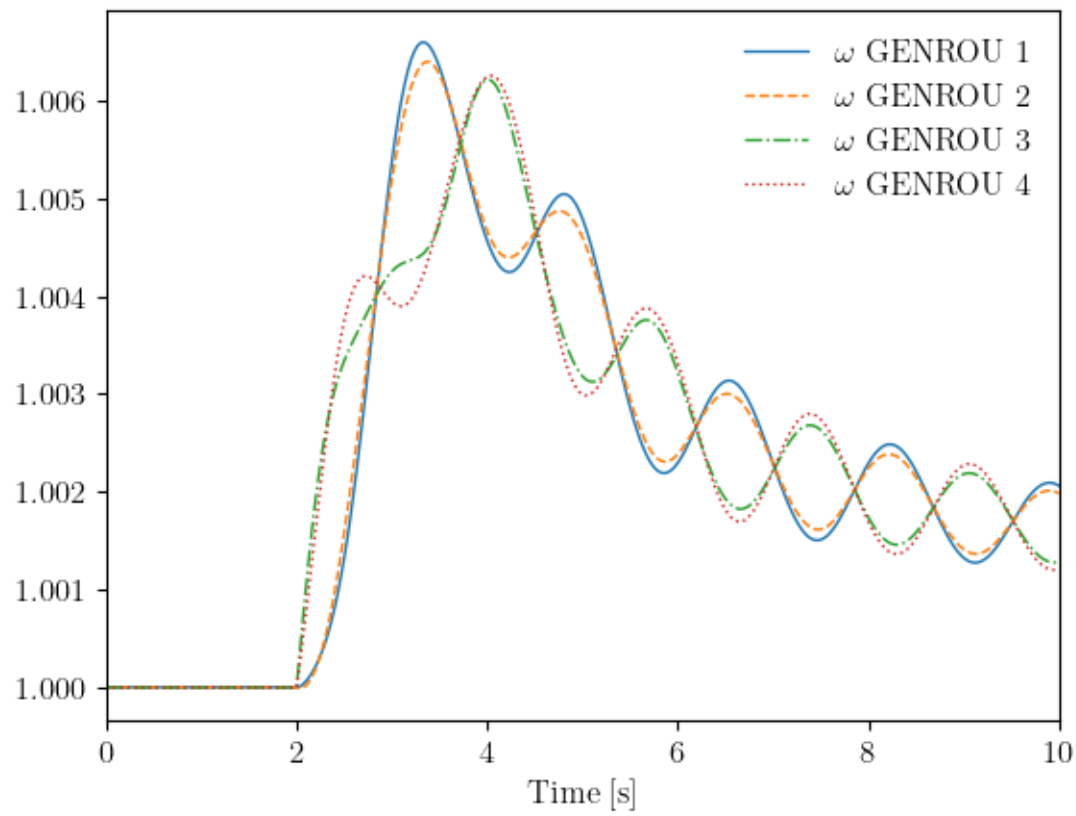
```
ss.TDS.load_plotter()
```

2.7.3 Plotting by Variable

The most straightforward way to plot simulation results is by passing a variable object directly to the `plot()` method. This approach is intuitive because you reference the exact quantity you want to visualize without needing to look up numerical indices.

For example, to plot all generator rotor speeds (the `omega` variable of the GENROU model), simply pass `ss.GENROU.omega` to the plotter.

```
fig, ax = ss.TDS.plt.plot(ss.GENROU.omega)
```



The plot shows the characteristic inter-area oscillations of the Kundur two-area system. After the line trips at t=2s, generators in the same area swing together while generators in different areas swing against each other. This inter-area mode is a key concern in power system stability analysis.

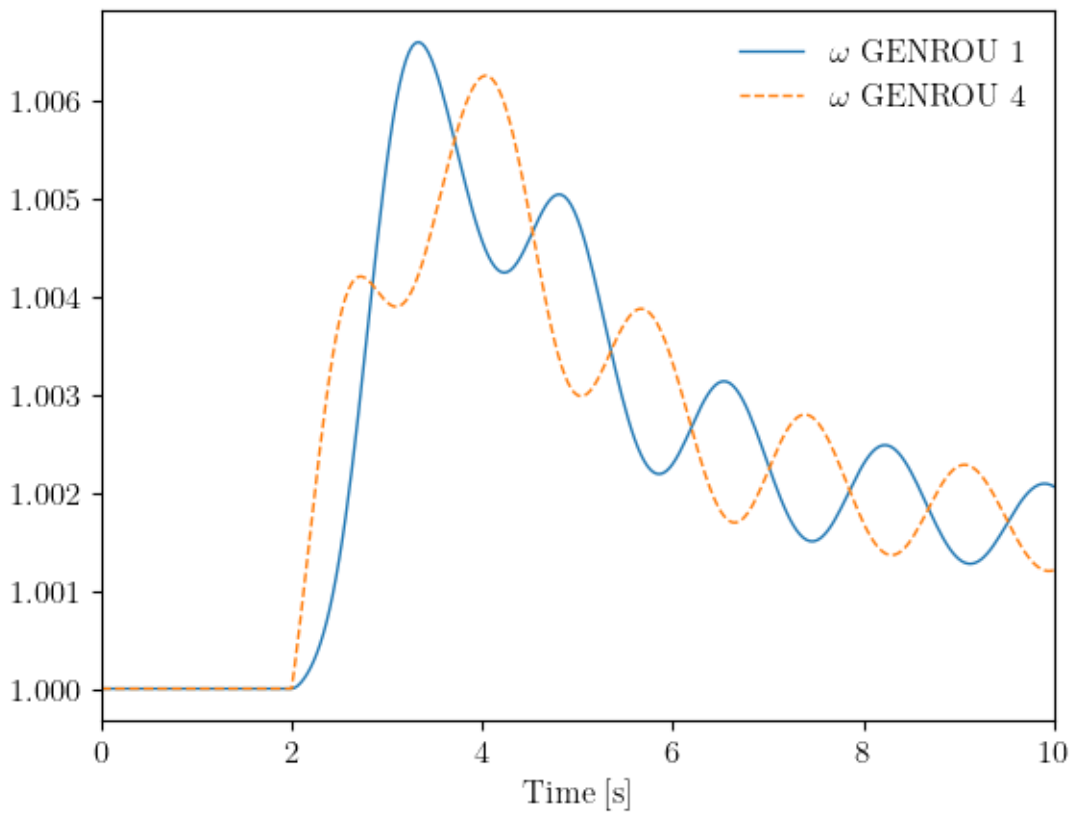
The `plot()` method returns both a figure object and an axis object, which you can use for further customization with matplotlib.

Plotting a Subset of Variables

When a model has many instances, you may only want to plot a subset rather than all of them. The `a` argument accepts a tuple of zero-indexed positions indicating which variables to include.

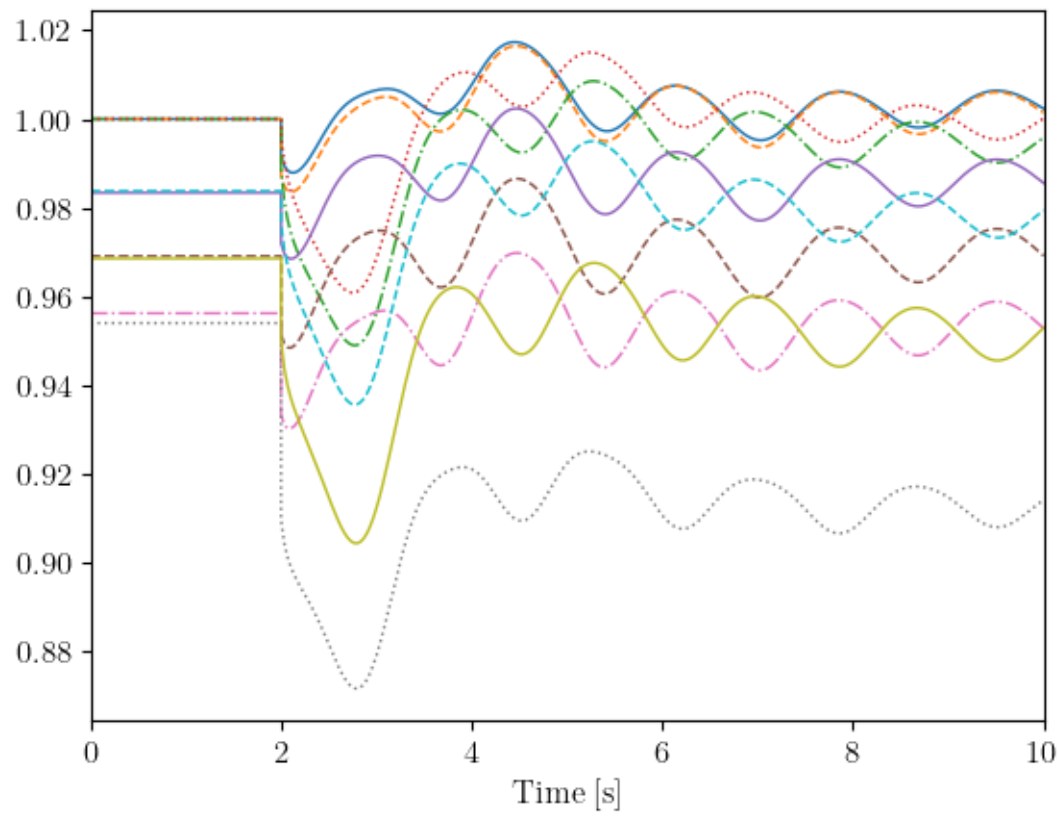
For example, to plot only the first and fourth generators (positions 0 and 3), use:

```
fig, ax = ss.TDS.plt.plot(ss.GENROU.omega, a=(0, 3))
```



You can also plot bus voltage magnitudes. The bus voltage variable `v` is accessed through `ss.Bus.v` (note that this is an algebraic variable rather than a state variable like generator speed).

```
fig, ax = ss.TDS.plt.plot(ss.Bus.v)
```



2.7.4 Index-Based Plotting

For more advanced use cases or when working with the command-line plotting tool, you may need to plot by variable index rather than by variable object. ANDES assigns each variable a unique index that is recorded in the output `.lst` file.

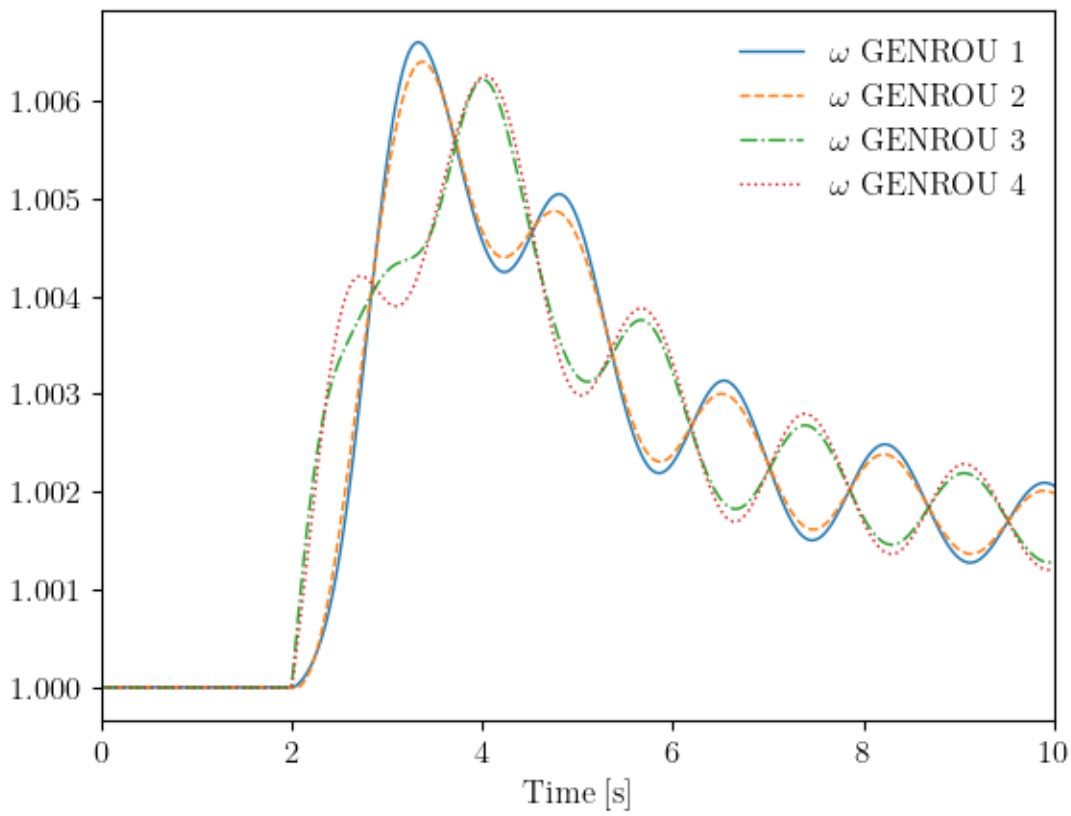
The `find()` method helps you discover these indices. It searches variable names using patterns (with regular expression support) and returns both the indices and the matching variable names.

```
# Find all variables containing 'omega'
ss.TDS.plt.find('omega')
```

```
([5, 6, 7, 8],
 ['omega GENROU 1', 'omega GENROU 2', 'omega GENROU 3', 'omega GENROU 4'])
```

Once you know the indices, you can pass them as a tuple or list to `plot()`. This produces the same result as plotting by variable but uses the numerical indices instead.

```
# Plot using indices directly
fig, ax = ss.TDS.plt.plot((5, 6, 7, 8))
```



The index-based approach is particularly useful when using the `andes plot` command from the terminal:

```
andes plot kundur_full_out.lst 0 5 6 7 8
```

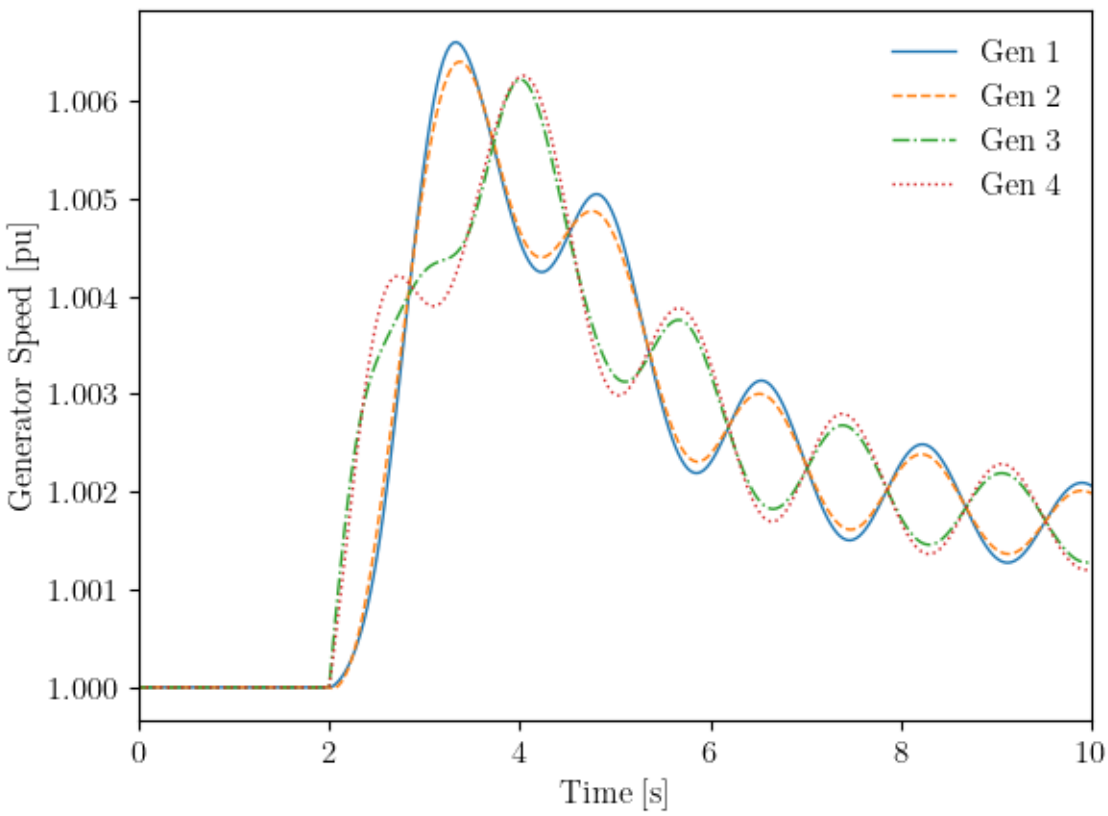
Here, index 0 represents time, and indices 5-8 represent the generator speeds.

2.7.5 Customizing Plots

The `plot()` method accepts numerous arguments for customizing the appearance of your figures. These options let you create publication-ready plots without needing to write additional matplotlib code.

Labels and Legends

```
fig, ax = ss.TDS.plt.plot(
    ss.GENROU.omega,
    ylabel='Generator Speed [pu]',
    yheader=['Gen 1', 'Gen 2', 'Gen 3', 'Gen 4']
)
```



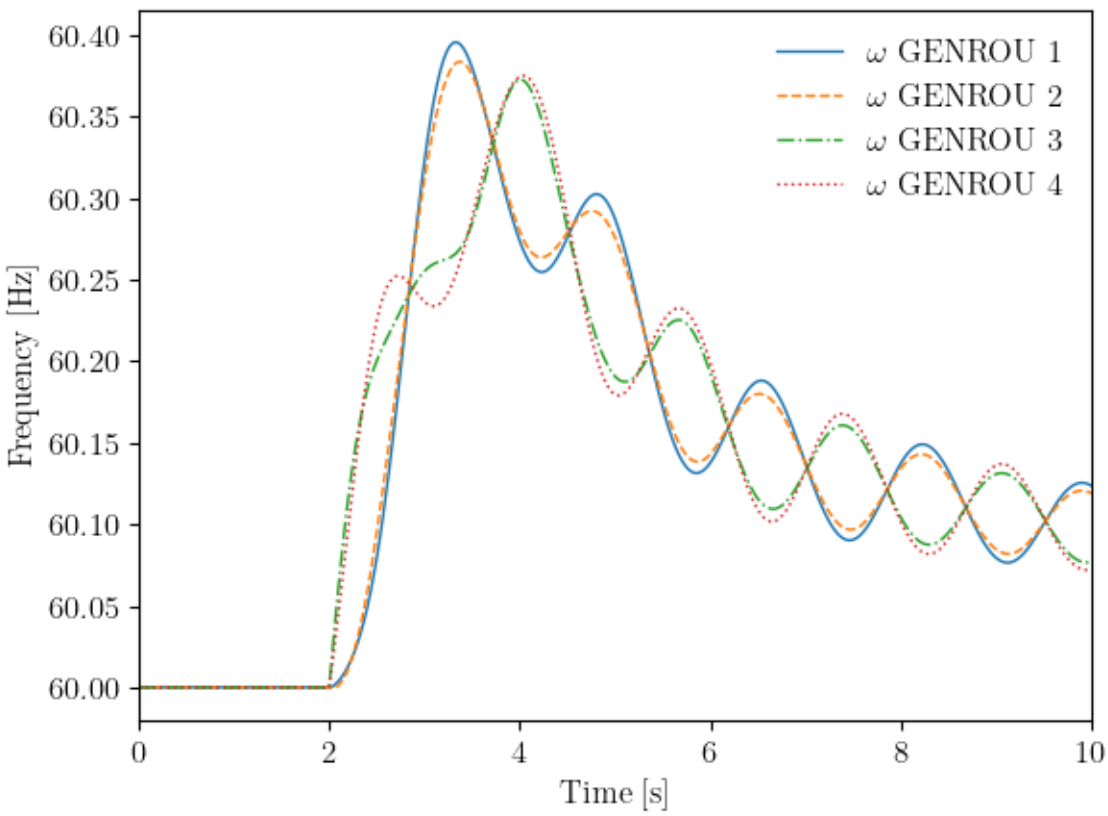
The `ylabel` argument sets the y-axis label, while `yheader` provides custom legend entries. When using `yheader`, ensure the number of entries matches the number of variables being plotted.

Scaling Values

Power system variables are often stored in per-unit but may be more intuitive when displayed in other units. The `ycalc` argument accepts a callable that transforms the y-values before plotting.

For example, to convert generator speed from per-unit to Hz (for a 60 Hz system where 1.0 pu = 60 Hz):

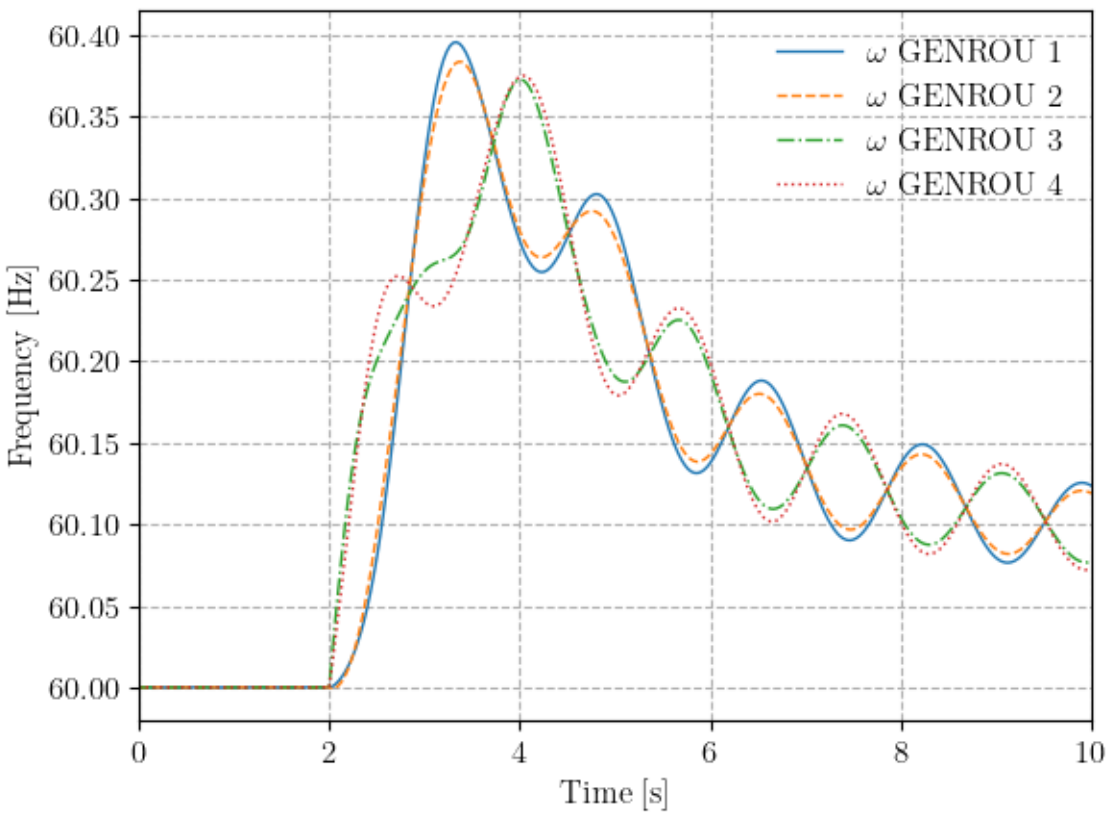
```
fig, ax = ss.TDS.plt.plot(
    ss.GENROU.omega,
    ycalc=lambda x: 60 * x,
    ylabel='Frequency [Hz]'
)
```



Grid and Greyscale

For technical publications or presentations, you may want to add a grid for easier reading or use greyscale for black-and-white printing.

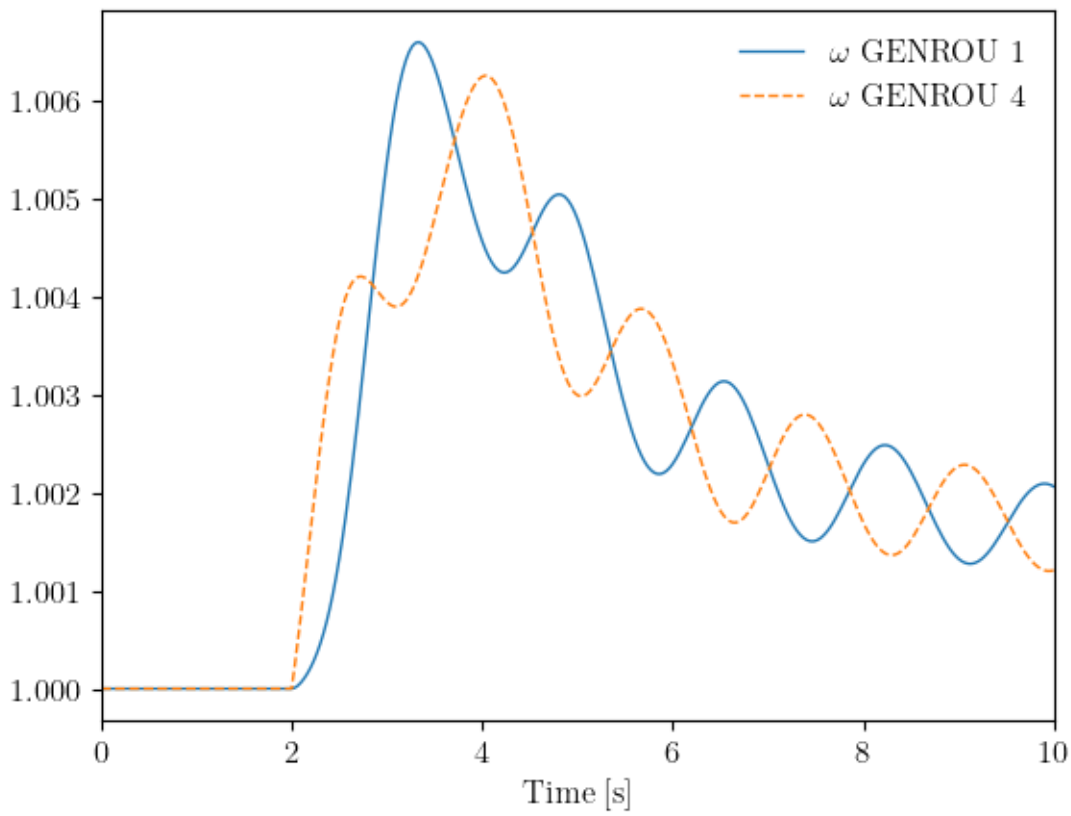
```
fig, ax = ss.TDS.plt.plot(
    ss.GENROU.omega,
    ycalc=lambda x: 60 * x,
    ylabel='Frequency [Hz]',
    grid=True,
    greyscale=True
)
```



Adding Curves to an Existing Figure

When comparing results from different scenarios or showing related variables on the same plot, you can add new curves to an existing figure. Pass the `fig` and `ax` objects from a previous plot to overlay new data.

```
# First plot: generators 1 and 4
fig, ax = ss.TDS.plt.plot(ss.GENROU.omega, a=(0, 3))

# Add generator 2 with a different line style
fig, ax = ss.TDS.plt.plot(
    ss.GENROU.omega,
    a=(1,),
    fig=fig,
    ax=ax,
    linestyles=['-.']
)
```




```
<Figure size 640x480 with 0 Axes>
```

2.7.6 Interactive Plots with Plotly

For exploratory analysis in Jupyter notebooks, ANDES supports an interactive Plotly backend. Plotly figures support zooming, panning, and hover tooltips that show exact values — all without any extra code.

To use it, pass `backend='plotly'` to the `plot()` method. The API is otherwise identical to the default matplotlib backend.

```
fig = ss.TDS.plt.plot(ss.GENROU.omega, backend='plotly')
```

All customization arguments (`ylabel`, `grid`, `ytimes`, `hline`, `vline`, axis limits, etc.) work the same way as with matplotlib. LaTeX legend labels are automatically replaced with plain text since Plotly does not render LaTeX in legends.

```
fig = ss.TDS.plt.plot(
    ss.GENROU.omega,
    ytimes=60,
    ylabel='Frequency [Hz]',
    title='Generator Frequencies',
    hline=60.0,
    grid=True,
```

(continues on next page)

(continued from previous page)

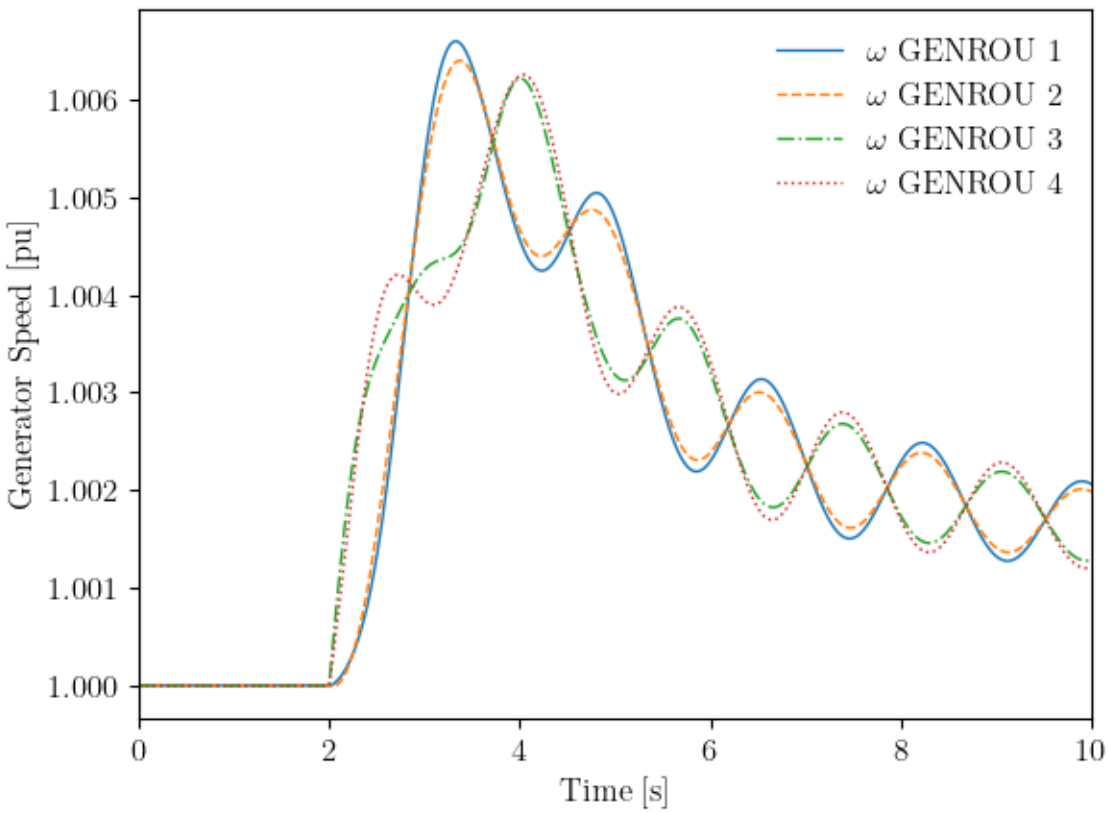
```
    backend='plotly',
)
```

2.7.7 Saving Figures

Figures can be saved directly from the `plot()` method using the `savefig` argument. When set to `True`, the figure is saved with an automatically generated filename. You can also pass a string to specify the exact filename.

```
fig, ax = ss.TDS.plt.plot(
    ss.GENROU.omega,
    ylabel='Generator Speed [pu]',
    savefig='generator_speeds.png'
)
```

```
Figure saved to "generator_speeds.png".
```



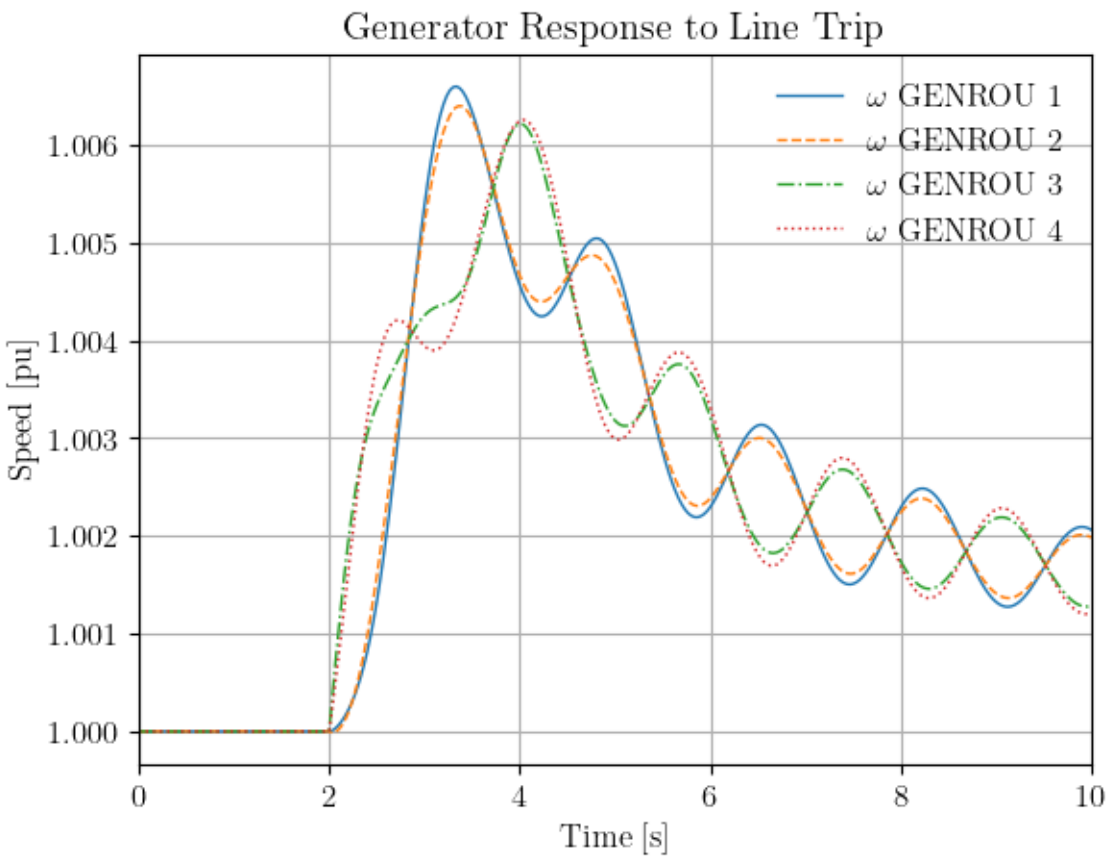
For more control over the output format and resolution, you can save the figure using matplotlib directly. This is useful when you need vector formats like PDF for publications.

```
fig, ax = ss.TDS.plt.plot(ss.GENROU.omega, show=False)
ax.set_title('Generator Response to Line Trip')
```

(continues on next page)

(continued from previous page)

```
ax.set_ylabel('Speed [pu]')
ax.grid(True)
fig.savefig('generator_response.pdf', dpi=300, bbox_inches='tight')
```



2.7.8 Exporting to CSV

While plots are useful for visualization, you often need the raw numerical data for further analysis in spreadsheets or other tools. The `export_csv()` method exports all simulation results to a CSV file that can be opened in Excel or read by other programs.

```
ss.TDS.plt.export_csv('simulation_results.csv')
```

```
CSV data saved to "simulation_results.csv".
```

```
'simulation_results.csv'
```

For more selective export, the `get_timeseries()` method returns a pandas DataFrame for any variable, ready for further analysis. For advanced use, the underlying arrays are accessible through `ss.dae.ts` (see *Time-Domain Simulation*).

```
# Get time series as DataFrames
omega_df = ss.TDS.get_timeseries(ss.GENROU.omega)
voltage_df = ss.TDS.get_timeseries(ss.Bus.v)

# Combine selected columns
import pandas as pd
df = pd.DataFrame({
    'Gen1_omega': omega_df.iloc[:, 0],
    'Gen2_omega': omega_df.iloc[:, 1],
    'Bus1_v': voltage_df.iloc[:, 0],
})
df.head()
```

```
          Gen1_omega  Gen2_omega  Bus1_v
0.000000         1.0         1.0     1.0
0.033333         1.0         1.0     1.0
0.066667         1.0         1.0     1.0
0.100000         1.0         1.0     1.0
0.133333         1.0         1.0     1.0
```

```
df.to_csv('selected_results.csv', index=False)
```

2.7.9 Plot Method Reference

The `plot()` method supports many additional arguments for fine-grained control. Here is a summary of the most commonly used options:

Argument	Description
`a`	Tuple of indices to select subset of variables
`ylabel`	Y-axis label
`yheader`	List of legend entries
`ycalc`	Callable to transform y-values
`grid`	Show grid lines (True/False)
`greyscale`	Use greyscale colors (True/False)
`savefig`	Save figure (True or filename)
`dpi`	Resolution for saved figure
`left`, `right`	X-axis limits
`ymin`, `ymax`	Y-axis limits
`fig`, `ax`	Existing figure/axis to plot on
`linestyles`	List of line styles
`line_width`	Width of plot lines
`font_size`	Size of labels and legends
`backend`	`'plotly'` for interactive Plotly plots, `None` for matplotlib (default)

2.7.10 Cleanup

```
!andes misc -C
!rm -f generator_speeds.png generator_response.pdf simulation_results.csv␣
↪selected_results.csv
```

```
"/home/docs/checkouts/readthedocs.org/user_builds/andes/checkouts/stable/docs/
↪source/tutorials/kundur_full_out.txt" removed.
"/home/docs/checkouts/readthedocs.org/user_builds/andes/checkouts/stable/docs/
↪source/tutorials/kundur_full_out.lst" removed.
"/home/docs/checkouts/readthedocs.org/user_builds/andes/checkouts/stable/docs/
↪source/tutorials/kundur_full_out.npz" removed.
```

2.7.11 Next Steps

- *Eigenvalue Analysis* - Small-signal stability analysis
- *Parameter Sweeps and Batch Processing* - Running batch simulations
- *Contingency Analysis* - N-1 contingency screening

2.8 Eigenvalue Analysis

Eigenvalue analysis (also called small-signal stability analysis) examines how a power system responds to small perturbations around its operating point. By linearizing the nonlinear differential-algebraic equations and computing the eigenvalues of the resulting state matrix, we can determine whether the system is stable and identify the oscillatory modes that dominate its dynamic behavior.

This analysis is fundamental to understanding inter-area oscillations, local plant modes, and control interactions. The location of eigenvalues in the complex plane tells us about stability: eigenvalues with negative real parts indicate stable modes (perturbations decay), while eigenvalues with positive real parts indicate unstable modes (perturbations grow). The imaginary part relates to the oscillation frequency.

This tutorial demonstrates how to run eigenvalue analysis in ANDES and interpret the results.

2.8.1 Setup

```
%matplotlib inline

import numpy as np
import andes

andes.config_logger(stream_level=20)
```

2.8.2 Running Eigenvalue Analysis

Eigenvalue analysis requires an established operating point, so ANDES automatically runs power flow before computing eigenvalues. The simplest way to run eigenvalue analysis is to pass `routine='eig'` to `andes.run()`, which loads the case, runs power flow, and then performs the eigenvalue calculation in a single step.

```
case_path = andes.get_case('kundur/kundur_full.xlsx')
ss = andes.run(case_path, routine='eig')
```

```
Working directory: "/home/docs/checkouts/readthedocs.org/user_builds/andes/
↪checkouts/stable/docs/source/tutorials"
> Loaded generated Python code in "/home/docs/.andes/pycode".
Parsing input file "/home/docs/checkouts/readthedocs.org/user_builds/andes/
↪envs/stable/lib/python3.11/site-packages/andes/cases/kundur/kundur_full.xlsx
↪"...
Input file parsed in 0.0313 seconds.
Connectivity check completed in 0.0001 seconds.
-> System connectivity check results:
  No islanded bus detected.
  System is interconnected.
  Each island has a slack bus correctly defined and enabled.
System internal structure set up in 0.0205 seconds.

-> Power flow calculation
           Numba: Off
   Sparse solver: KLU
 Solution method: NR method
Power flow initialized in 0.0031 seconds.
0: |F(x)| = 14.9282832
1: |F(x)| = 3.608627841
2: |F(x)| = 0.1701107882
3: |F(x)| = 0.002038626956
4: |F(x)| = 3.745103977e-07
Converged in 5 iterations in 0.0025 seconds.
Report saved to "kundur_full_out.txt" in 0.0008 seconds.
DAE compaction: removed 5 algebraic variable slots (m: 149 -> 144)
Initialization for dynamics completed in 0.0152 seconds.
Initialization was successful.

-> Eigenvalue Analysis:
  Positive       0
  Zeros          1
  Negative      52
Eigenvalue analysis finished in 0.0018 seconds.
Report saved to "kundur_full_eig.txt".
-> Single process finished in 0.6308 seconds.
```

The output shows that the system has 52 negative eigenvalues, 1 zero eigenvalue, and 0 positive eigenvalues. The zero eigenvalue is expected and corresponds to the rotor angle reference (absolute angles are not observable, only relative angles matter). The absence of positive eigenvalues indicates that the system is stable for small perturbations.

Alternatively, you can run eigenvalue analysis step by step after loading the system:

```
# Step-by-step approach
ss2 = andes.load(andes.get_case('kundur/kundur_full.xlsx'))
ss2.PFlow.run()
ss2.EIG.run()
```

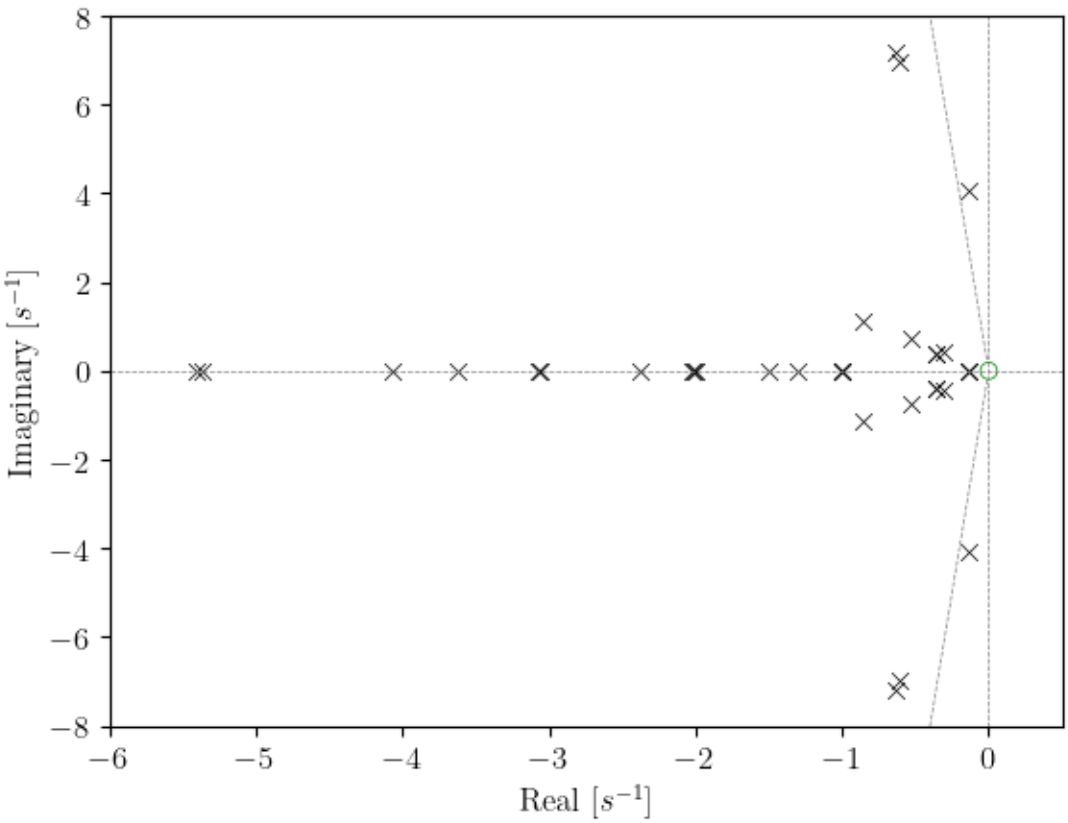
```
Working directory: "/home/docs/checkouts/readthedocs.org/user_builds/andes/
↪checkouts/stable/docs/source/tutorials"
> Reloaded generated Python code of module "pycode".
Parsing input file "/home/docs/checkouts/readthedocs.org/user_builds/andes/
↪envs/stable/lib/python3.11/site-packages/andes/cases/kundur/kundur_full.xlsx
↪"...
Input file parsed in 0.0305 seconds.
Connectivity check completed in 0.0001 seconds.
-> System connectivity check results:
  No islanded bus detected.
  System is interconnected.
  Each island has a slack bus correctly defined and enabled.
System internal structure set up in 0.0189 seconds.

-> Power flow calculation
           Numba: Off
   Sparse solver: KLU
 Solution method: NR method
Power flow initialized in 0.0030 seconds.
0: |F(x)| = 14.9282832
1: |F(x)| = 3.608627841
2: |F(x)| = 0.1701107882
3: |F(x)| = 0.002038626956
4: |F(x)| = 3.745103977e-07
Converged in 5 iterations in 0.0024 seconds.
Report saved to "kundur_full_out.txt" in 0.0005 seconds.
DAE compaction: removed 5 algebraic variable slots (m: 149 -> 144)
Initialization for dynamics completed in 0.0149 seconds.
Initialization was successful.

-> Eigenvalue Analysis:
  Positive       0
  Zeros          1
  Negative      52
Eigenvalue analysis finished in 0.0015 seconds.
Report saved to "kundur_full_eig.txt".
```

```
True
```

2.8.3 Plotting Eigenvalues on the S-Plane

The eigenvalue results are most easily interpreted through visualization. The `ss.EIG.plot()` method creates an s-plane plot where the x-axis represents the real part (damping) and the y-axis represents the imaginary part (oscillation frequency in rad/s).

```
ss.EIG.plot()
```




```
(<Figure size 640x480 with 1 Axes>,
 <Axes: xlabel='Real [$s^{-1}$]', ylabel='Imaginary [$s^{-1}$]'>)
```

In the plot, eigenvalues appearing on the left half of the plane (negative real part) represent stable modes. Those closer to the imaginary axis are less damped and will decay slowly after a disturbance. Eigenvalues with larger imaginary parts correspond to higher-frequency oscillations.

For the Kundur two-area system, you can identify several types of modes:

- Modes near the origin with small imaginary parts are often inter-area oscillation modes
- Modes farther from the origin typically correspond to local generator modes and exciter/governor dynamics
- Real eigenvalues (on the real axis) represent non-oscillatory modes

2.8.4 Eigenvalue Report

ANDES generates a detailed report file (`*_eig.txt`) containing the computed eigenvalues, their damping ratios, and frequencies. This report is useful for detailed analysis and documentation.

```
with open('kundur_full_eig.txt', 'r') as f:
    content = f.read()
    # Print first 2000 characters to see the structure
    print(content[:2000])
```

```
ANDES 2.0.0
Copyright (C) 2015-2026 Hantao Cui

ANDES comes with ABSOLUTELY NO WARRANTY
Case file: /home/docs/checkouts/readthedocs.org/user_builds/andes/envs/stable/
↪lib/python3.11/site-packages/andes/cases/kundur/kundur_full.xlsx
Report time: 03/12/2026 05:07:07 PM

Power flow converged in 5 iterations.
Flat-start: No


EIGENVALUE ANALYSIS REPORT

Positives
Zeros
Negatives

STATISTICS
                      Most Associated              Real              Imag.      ␣
↪Damped Freq.         Frequency       Damping [%]

#1                      LL_x EXDC2 1                -1                  0      ␣
↪           0                 0                 0
#2                      LL_x EXDC2 2                -1                  0      ␣
↪           0                 0                 0
#3                      LL_x EXDC2 3                -1                  0      ␣
↪           0                 0                 0
#4                      LL_x EXDC2 4                -1                  0      ␣
↪           0                 0                 0
#5                      LS_y EXDC2 1           -49.541                  0      ␣
↪           0                 0                 0
#6                      LS_y EXDC2 1           -49.526                  0      ␣
↪           0                 0                 0
#7                      LA_y EXDC2 3           -49.199             0.4645      ␣
↪    0.073927            7.8306            99.996
#8                      LA_y EXDC2 3           -49.199            -0.4645      ␣
↪    0.073927            7.8306            99.996
#9                      LS_y EXDC2 1           -49.203            0.34143      ␣
↪     0.05434            7.8312            99.998
#10                     LS_y EXDC2 1           -49.203           -0.34143      ␣
↪     0.05434            7.8312            99.998
```

(continues on next page)

(continued from previous page)

```
#11                        LA_y EXDC2 1              -48.88                    0      ␣
↪            0                   0                   0
#12
```

2.8.5 Parameter Sweeps and Root Loci

One of the most powerful applications of eigenvalue analysis is studying how eigenvalues move as system parameters change. This root locus analysis helps identify parameter ranges that maintain stability and shows which modes are most sensitive to parameter variations.

The `ss.EIG.sweep()` method automates this process by repeatedly computing eigenvalues for different parameter values. In this example, we study how the exciter gain `KA` of the first EXDC2 exciter affects system stability by varying it from 20 to 200.

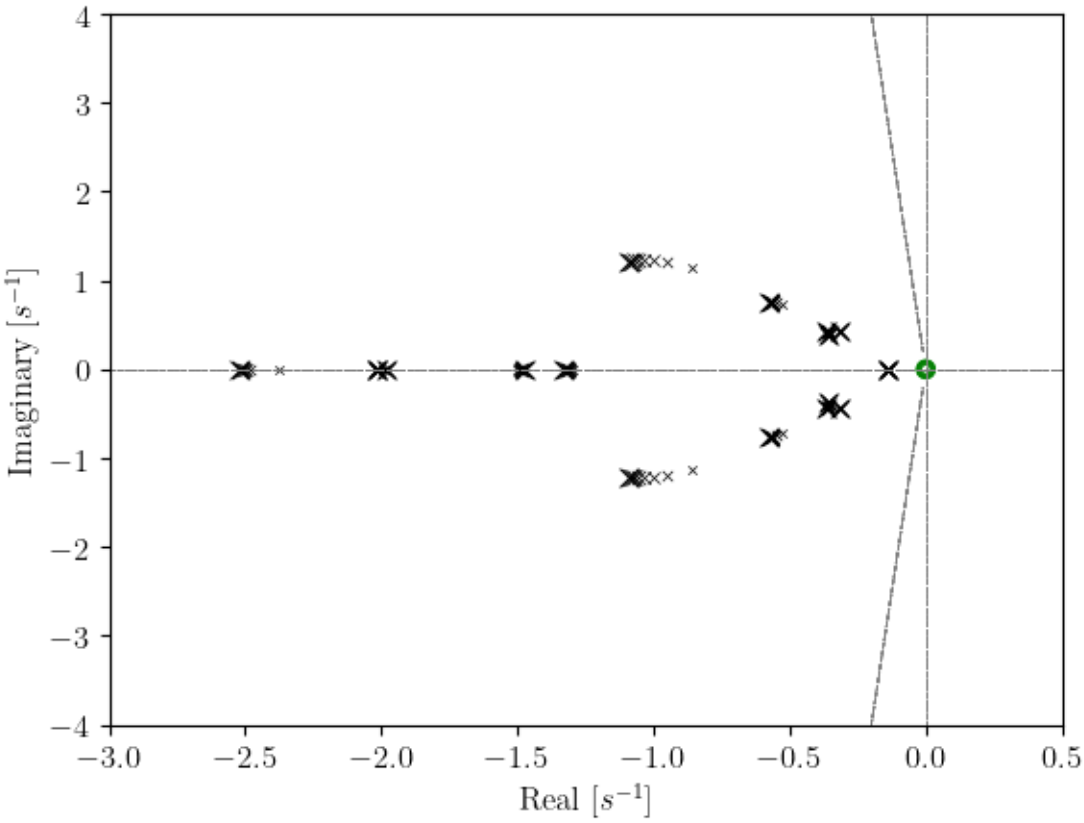
```
# Sweep KA from 20 to 200 in 10 steps
ret = ss.EIG.sweep(ss.EXDC2.KA, 1, np.linspace(20, 200, 10))
```

The `sweep()` method returns a dictionary containing the eigenvalues computed at each parameter value. We can then visualize how specific eigenvalues move using `plot_root_loci()`.

The following plot shows eigenvalues 30 through 50 (zero-indexed). The marker size increases with the parameter value, so larger markers correspond to higher exciter gains. This allows you to track the trajectory of each eigenvalue as the parameter increases.

```
ss.EIG.plot_root_loci(ret, range(30, 51), left=-3, ymax=4, ymin=-4)
```

```
(<Figure size 640x480 with 1 Axes>,
 <Axes: xlabel='Real [$s^{-1}$]', ylabel='Imaginary [$s^{-1}$]'>)
```



The root locus plot reveals how increasing the exciter gain affects system modes. If eigenvalues move toward the right (toward the imaginary axis or beyond), the system becomes less stable or potentially unstable. If they move left, the system becomes more damped.

This type of analysis is invaluable for tuning controller parameters. By observing the root loci, you can identify the parameter range that provides acceptable damping while avoiding instability.

2.8.6 Cleanup

```
!andes misc -C
```

```
"/home/docs/checkouts/readthedocs.org/user_builds/andes/checkouts/stable/docs/
↪source/tutorials/kundur_full_eig.txt" removed.
"/home/docs/checkouts/readthedocs.org/user_builds/andes/checkouts/stable/docs/
↪source/tutorials/kundur_full_out.txt" removed.
```

2.8.7 Next Steps

- *Parameter Sweeps and Batch Processing* - Batch simulations for sensitivity studies
- *Contingency Analysis* - N-1 contingency screening
- *Dynamic Control and Setpoint Changes* - Runtime parameter control

2.9 Parameter Sweeps and Batch Processing

Power system studies frequently require running many simulations with systematically varied parameters. You might want to understand how load levels affect voltage stability, how different fault locations impact system response, or how controller tuning parameters influence dynamic behavior. Rather than manually running each scenario one at a time, ANDES provides several approaches for batch processing that can dramatically reduce the time required for comprehensive studies.

This tutorial covers three approaches to batch simulation: file-based parallel processing for large studies, in-memory loops for smaller studies, and the `pool` option for intermediate cases where you want both parallelism and programmatic access to results.

Note: Prerequisites: This tutorial assumes familiarity with Python fundamentals including loops, dictionaries, and NumPy arrays. Complete *First Simulation* through *Eigenvalue Analysis* before proceeding.

2.9.1 Setup

```
%matplotlib inline

import andes
import numpy as np
from matplotlib import pyplot as plt
import os
import shutil

andes.config_logger(stream_level=30)  # Reduce logging verbosity for batch␣
↪runs
```

2.9.2 File-Based Batch Processing

For large parametric studies involving hundreds or thousands of scenarios, the most efficient approach is to generate separate case files and then run them in parallel using the ANDES command-line interface. This approach leverages multi-core processors effectively and allows you to checkpoint progress by checking which output files exist.

Generating Case Files

The workflow begins by loading a base case, modifying parameters programmatically, and saving each variation to a new file. In this example, we create three cases with load levels varying from 100% to 120% of the base value.

```
# Create output directory for the case files
os.makedirs('batch_cases', exist_ok=True)

# Load base case
kundur = andes.get_case('kundur/kundur_full.xlsx')
ss = andes.load(kundur)
```

```
# Get the base load value
p0_base = ss.PQ.get('p0', 'PQ_0')
print(f"Base load: {p0_base:.2f} MW")

# Create cases with load varying from 100% to 120% of base
N_CASES = 3
p0_values = np.linspace(p0_base, 1.2 * p0_base, N_CASES)

for value in p0_values:
    ss.PQ.set('p0', 'PQ_0', value, base='device')
    file_name = f'batch_cases/kundur_p_{value:.2f}.xlsx'
    andes.io.dump(ss, 'xlsx', file_name, overwrite=True)
    print(f"Created: {file_name}")
```

```
Base load: 11.59 MW
Created: batch_cases/kundur_p_11.59.xlsx
Created: batch_cases/kundur_p_12.75.xlsx
Created: batch_cases/kundur_p_13.91.xlsx
```

Running Cases in Parallel

Once the case files are generated, you can run all of them in parallel using the `andes run` command with wildcards. ANDES automatically detects the number of CPU cores and distributes the workload across them.

```
# Run all cases with time-domain simulation
!andes run batch_cases/*.xlsx -r tds
```

```

    _           _         | Version 2.0.0
   /_\  _ _  __| |___ ___ | Python 3.11.12 on Linux, 03/12/2026 05:07:15 PM
  / _ \| ' \/ _` / -_|_-< |
 /_/ \_\_||_\__,_\___/__/ | This program comes with ABSOLUTELY NO WARRANTY.

Working directory: "/home/docs/checkouts/readthedocs.org/user_builds/andes/
↪checkouts/stable/docs/source/tutorials"
> Loaded generated Python code in "/home/docs/.andes/pycode".
```

(continues on next page)

(continued from previous page)

```
-> Processing 3 jobs on 2 CPUs.
  0%|                                                          | 0/100 [00:00<?, ?%/
↪s]
 13%|#####2                                  | 13.0/100 [00:00<00:00, 128.87%/
↪s]
 26%|##########6                              | 26.0/100 [00:00<00:00, 94.59%/
↪s]
 37%|###############1                        | 37.0/100 [00:00<00:00, 93.07%/
↪s]
 47%|###################2                     | 47.0/100 [00:00<00:00, 90.19%/
↪s]
 57%|#######################3                 | 57.0/100 [00:00<00:00, 88.56%/
↪s]
 66%|###########################              | 66.0/100 [00:00<00:00, 88.13%/
↪s]
 74%|##############################3          | 74.0/100 [00:00<00:00, 87.10%/
↪s]
 76%|################################1        | 76.0/100 [00:00<00:00, 88.32%/
↪s]
 83%|##################################       | 83.0/100 [00:00<00:00, 87.31%/
↪s]
 86%|####################################2     | 86.0/100 [00:00<00:00, 89.90%/
↪s]
 93%|######################################1  | 93.0/100 [00:01<00:00, 89.31%/
↪s]
100%|########################################| 100.0/100 [00:01<00:00, 92.31%/
↪s]
100%|########################################| 100.0/100 [00:01<00:00, 90.66%/
↪s]
  0%|                                                          | 0/100 [00:00<?, ?%/
↪s]
 13%|#####2                                  | 13.0/100 [00:00<00:00, 126.29%/
↪s]
 26%|##########6                              | 26.0/100 [00:00<00:00, 95.48%/
↪s]
 36%|##############7                         | 36.0/100 [00:00<00:00, 91.85%/
↪s]
 46%|##################8                      | 46.0/100 [00:00<00:00, 90.42%/
↪s]
 56%|######################9                  | 56.0/100 [00:00<00:00, 89.83%/
↪s]
 66%|###########################              | 66.0/100 [00:00<00:00, 89.95%/
↪s]
 76%|################################1        | 76.0/100 [00:00<00:00, 91.46%/
↪s]
 87%|#####################################6     | 87.0/100 [00:00<00:00, 95.01%/
↪s]
100%|########################################| 100.0/100 [00:01<00:00, 94.78%/
↪s]
Log saved to "/tmp/andes-r0drvm2m/andes.log".
-> Multiprocessing finished in 3.5784 seconds.
```

If you need to limit CPU usage (for example, to leave resources for other tasks on a shared workstation), use the `--ncpu` flag to specify the maximum number of parallel processes.

```
# Limit to 2 parallel processes
!andes run batch_cases/*.xlsx -r tds --ncpu 2
```

```
    _           _         | Version 2.0.0
   /_\  _ _  __| |___ ___ | Python 3.11.12 on Linux, 03/12/2026 05:07:19 PM
  / _ \| ' \/ _` / -_|_-< |
 /_/ \_\_||_\__,_\___/__/ | This program comes with ABSOLUTELY NO WARRANTY.

Working directory: "/home/docs/checkouts/readthedocs.org/user_builds/andes/
↪checkouts/stable/docs/source/tutorials"
> Loaded generated Python code in "/home/docs/.andes/pycode".
-> Processing 3 jobs on 2 CPUs.
  0%|                                                    | 0/100 [00:00<?, ?%/
↪s]
  0%|                                                    | 0/100 [00:00<?, ?%/
↪s]
 13%|#####2                                  | 13.0/100 [00:00<00:00, 127.61%/
↪s]
 13%|#####2                                  | 13.0/100 [00:00<00:00, 125.89%/
↪s]
 26%|##########6                              | 26.0/100 [00:00<00:00, 93.32%/
↪s]
 36%|##############7                          | 36.0/100 [00:00<00:00, 90.22%/
↪s]
 46%|##################8                      | 46.0/100 [00:00<00:00, 89.12%/
↪s]
 46%|##################8                      | 46.0/100 [00:00<00:00, 87.10%/
↪s]
 55%|######################5                  | 55.0/100 [00:00<00:00, 86.06%/
↪s]
 64%|##########################2              | 64.0/100 [00:00<00:00, 85.58%/
↪s]
 73%|##############################9          | 73.0/100 [00:00<00:00, 86.01%/
↪s]
 82%|##################################6      | 82.0/100 [00:00<00:00, 86.58%/
↪s]
 92%|######################################7  | 92.0/100 [00:01<00:00, 88.11%/
↪s]
100%|########################################| 100.0/100 [00:01<00:00, 90.02%/
↪s]
100%|########################################| 100.0/100 [00:01<00:00, 89.33%/
↪s]
  0%|                                                    | 0/100 [00:00<?, ?%/
↪s]
 13%|#####2                                  | 13.0/100 [00:00<00:00, 126.47%/
↪s]
 26%|##########6                              | 26.0/100 [00:00<00:00, 95.24%/
↪s]
 36%|##############7                          | 36.0/100 [00:00<00:00, 91.56%/
```

(continues on next page)

(continued from previous page)

```
↪s]
 46%|##################8                     | 46.0/100 [00:00<00:00, 90.22%/
↪s]
 56%|######################9                 | 56.0/100 [00:00<00:00, 89.64%/
↪s]
 66%|###########################             | 66.0/100 [00:00<00:00, 89.74%/
↪s]
 76%|################################1       | 76.0/100 [00:00<00:00, 91.39%/
↪s]
 87%|####################################6   | 87.0/100 [00:00<00:00, 94.51%/
↪s]
 98%|########################################1| 98.0/100 [00:01<00:00, 97.00%/
↪s]
100%|#########################################| 100.0/100 [00:01<00:00, 94.55%/
↪s]
Log saved to "/tmp/andes-oqy0sg97/andes.log".
-> Multiprocessing finished in 3.6100 seconds.
```

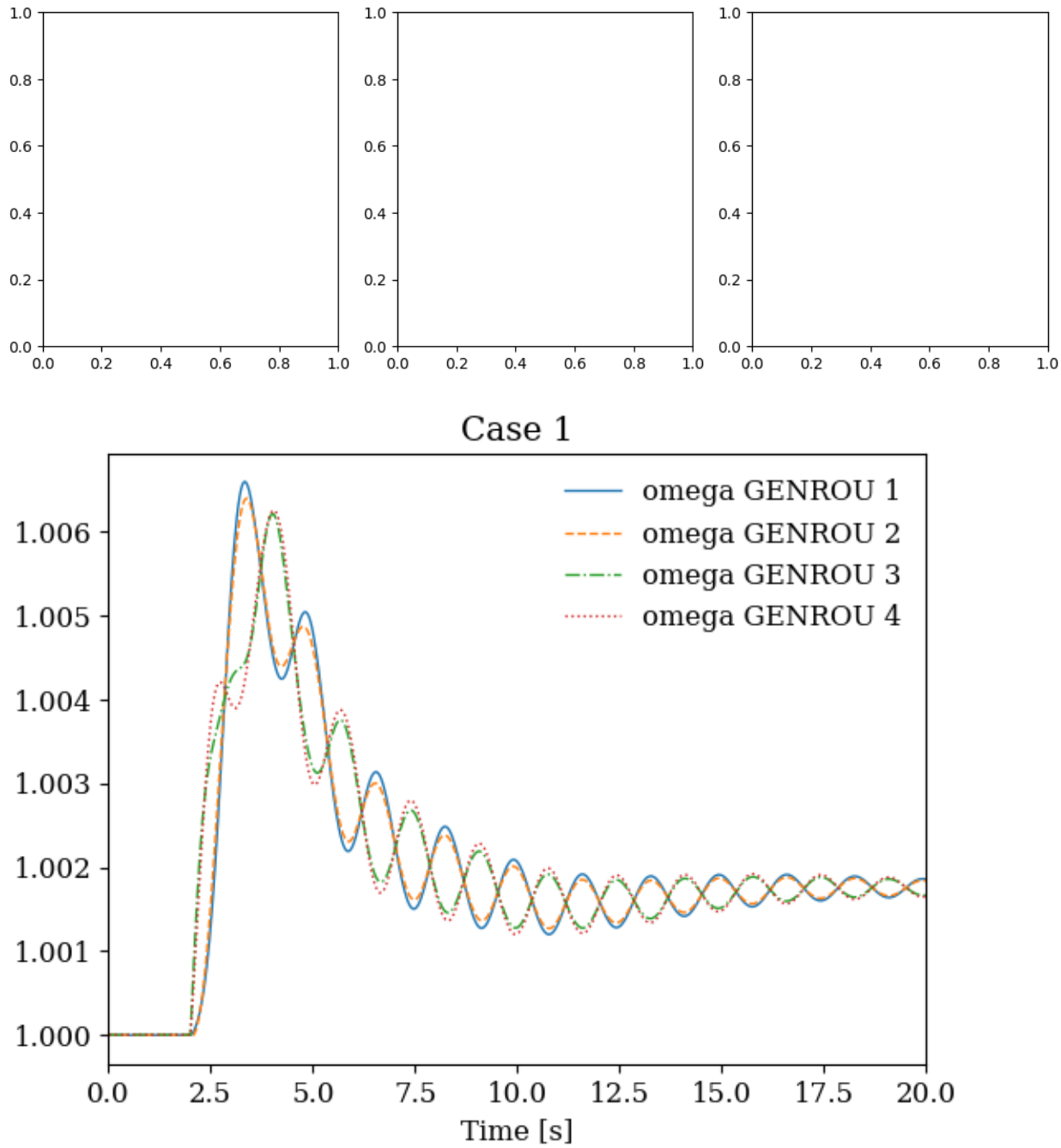
Returning System Objects for Post-Processing

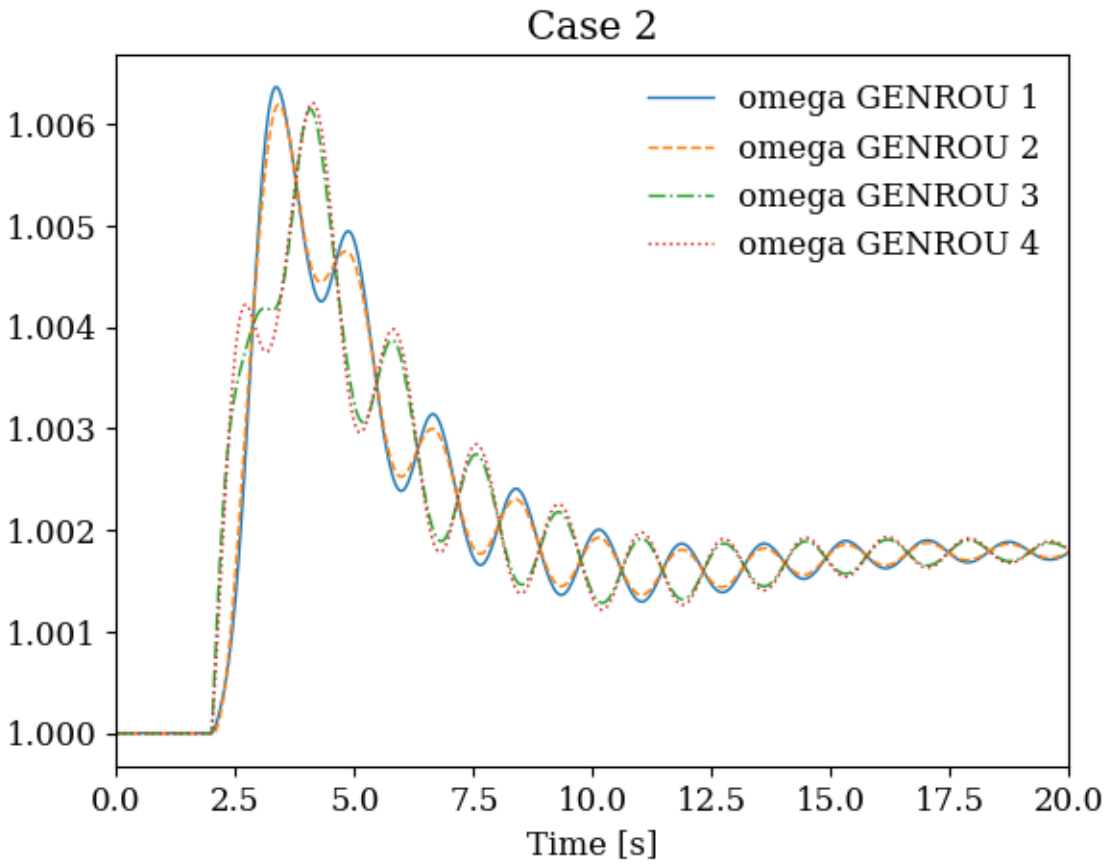
When you want to analyze results programmatically after batch execution, use `pool=True` in the Python API. This runs cases in parallel and returns a list of System objects that you can then analyze or plot.

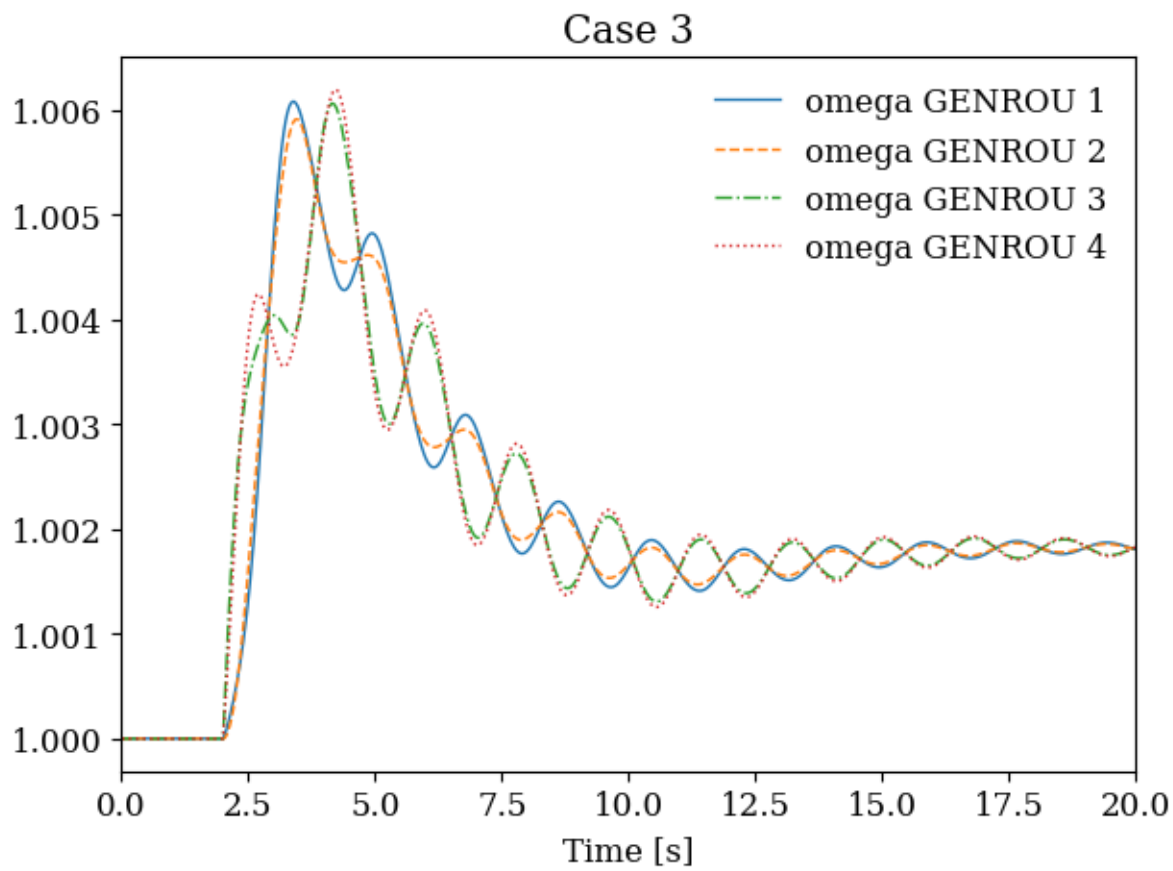
```
# Run all cases and return System objects
systems = andes.run('batch_cases/*.xlsx', routine='tds', pool=True,␣
↪verbose=30)

print(f"Completed {len(systems)} simulations")
print(f"Type of each result: {type(systems[0]).__name__}")
```

```
Completed 3 simulations
Type of each result: System
```

```
# Plot results from all cases side by side
fig, axes = plt.subplots(1, len(systems), figsize=(12, 4))

for i, sys in enumerate(systems):
    sys.TDS.plt.plot(sys.GENROU.omega, ax=axes[i],
                     title=f'Case {i+1}', latex=False, show=False)

plt.tight_layout()
plt.show()
```


Case 1
omega GENROU 1
omega GENROU 2
omega GENROU 3
omega GENROU 4
1.006
1.005
1.004
1.003
1.002
1.001
1.000
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Time [s]


Case 2
omega GENROU 1
omega GENROU 2
omega GENROU 3
omega GENROU 4
1.006
1.005
1.004
1.003
1.002
1.001
1.000
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Time [s]



2.9.3 In-Memory Parameter Sweeps

For smaller studies where generating files would be unnecessary overhead, you can simply loop through parameter values in Python and accumulate results. This approach is particularly useful for quick exploratory analysis during model development or for sweeps involving only a few cases.

Power Flow Parameter Sweep

The following example sweeps load levels and records the resulting bus voltage profiles. Since power flow converges quickly, this type of sweep can complete in seconds even for many parameter values.

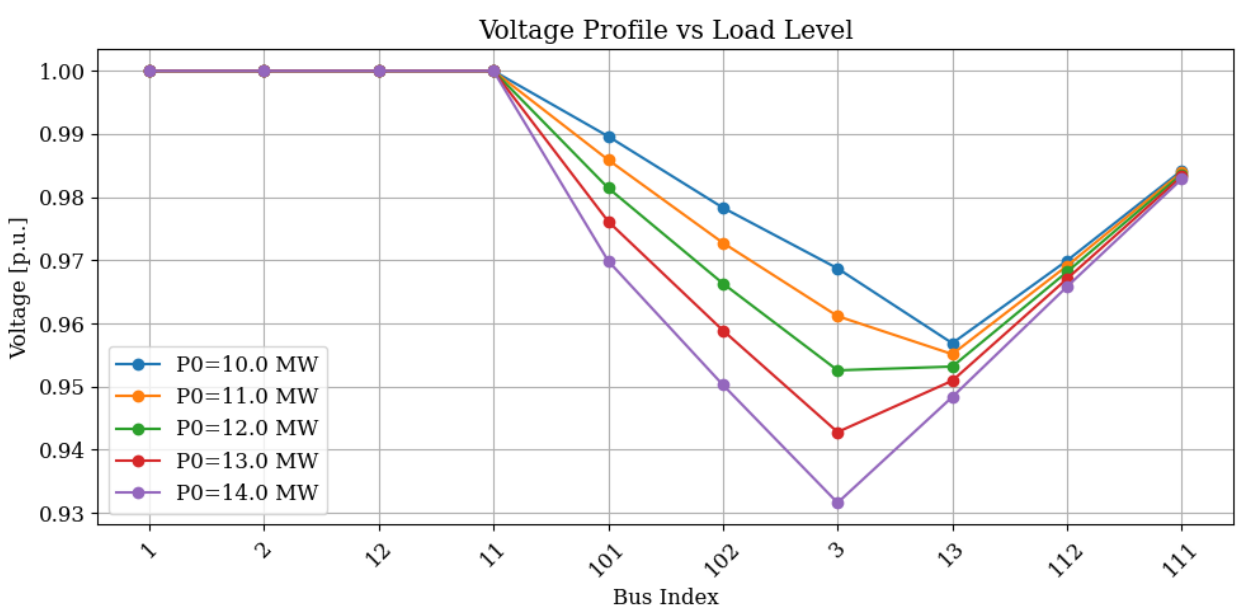
```
# Load system
ss = andes.run(kundur, no_output=True, default_config=True, verbose=30)

# View current load parameters
ss.PQ.as_df(vin=True)
```

```
      idx    u  name  bus     Vn     p0     q0  vmax  vmin  owner
uid
0    PQ_0  1.0  PQ_0    7  230.0  11.59 -0.735   1.1   0.9      1
1    PQ_1  1.0  PQ_1    8  230.0  15.75 -0.899   1.1   0.9      1
```

```
# Define parameter sweep range
n_samples = 5
pq0_values = np.linspace(10, 14, n_samples)  # MW range for PQ_0

# Storage for results: voltage at each bus for each parameter value
v_results = np.zeros((ss.Bus.n, n_samples))

# Run power flow for each load level
for i, p0 in enumerate(pq0_values):
    ss.PQ.set('p0', 'PQ_0', p0, base='device')
    ss.PFlow.run()
    v_results[:, i] = ss.dae.y[ss.Bus.v.a]

print(f"Completed {n_samples} power flow calculations")
```

```
Completed 5 power flow calculations
```

```
# Visualize how voltage profiles change with load level
plt.figure(figsize=(10, 5))
for i in range(n_samples):
    plt.plot(v_results[:, i], 'o-', label=f'P0={pq0_values[i]:.1f} MW')

plt.xlabel('Bus Index')
plt.ylabel('Voltage [p.u.]')
plt.title('Voltage Profile vs Load Level')
plt.legend()
plt.xticks(range(ss.Bus.n), ss.Bus.name.v, rotation=45)
plt.grid(True)
plt.tight_layout()
plt.show()
```



Batch Time-Domain Simulation

The same looping approach works for time-domain simulations, though each iteration takes longer. This example demonstrates running multiple contingency scenarios with different line trips. For each scenario, we create a fresh System object, add the appropriate disturbance, run the simulation, and store the result.

```
# Find available lines for contingency analysis
ss = andes.load(kundur, setup=False)
print(f"Available lines: {ss.Line.idx.v}")

# Select a subset for demonstration
lines_to_test = ss.Line.idx.v[:3]
```

```
Available lines: ['Line_0', 'Line_1', 'Line_2', 'Line_3', 'Line_4', 'Line_5',
↪'Line_6', 'Line_7', 'Line_8', 'Line_9', 'Line_10', 'Line_11', 'Line_12',
↪'Line_13', 'Line_14']
```

```
results = dict()

for line_idx in lines_to_test:
    # Load fresh system with setup=False to allow adding devices
    ss = andes.load(kundur, setup=False)

    # Add line trip at t=1.0s, restore at t=1.1s
    ss.add('Toggle', model='Line', dev=line_idx, t=1.0)
    ss.add('Toggle', model='Line', dev=line_idx, t=1.1)

    # Finalize system setup
    ss.setup()

    # Disable any existing Toggle from the test case
    ss.Toggle.set_status(1, 0)

    # Run simulation
    ss.PFlow.run()
    ss.TDS.config.tf = 5
    ss.TDS.config.no_tqdm = 1
    ss.TDS.run()

    results[line_idx] = ss
    print(f"Completed: {line_idx} trip")
```

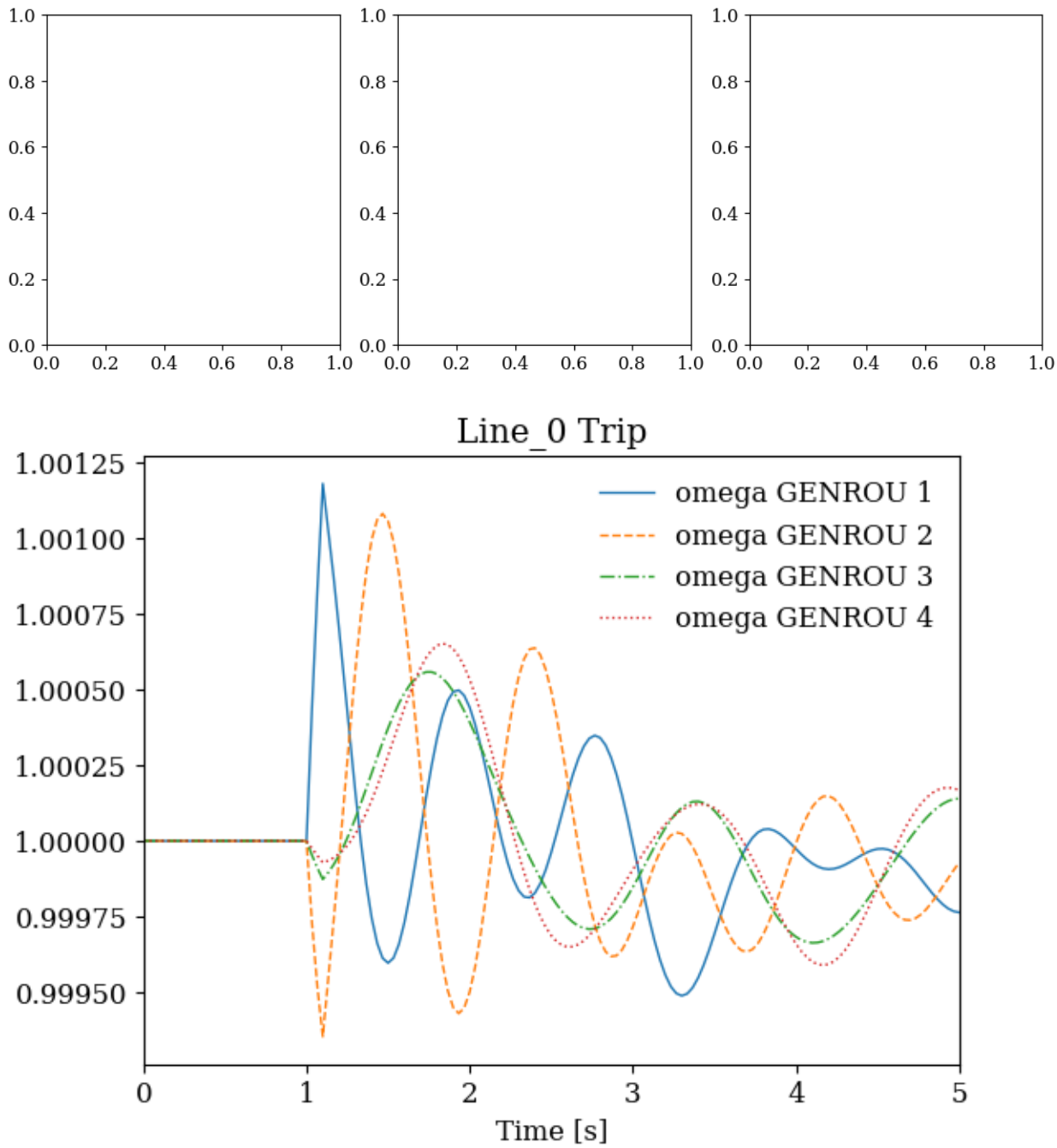
```
Completed: Line_0 trip
Completed: Line_1 trip
Completed: Line_2 trip
```

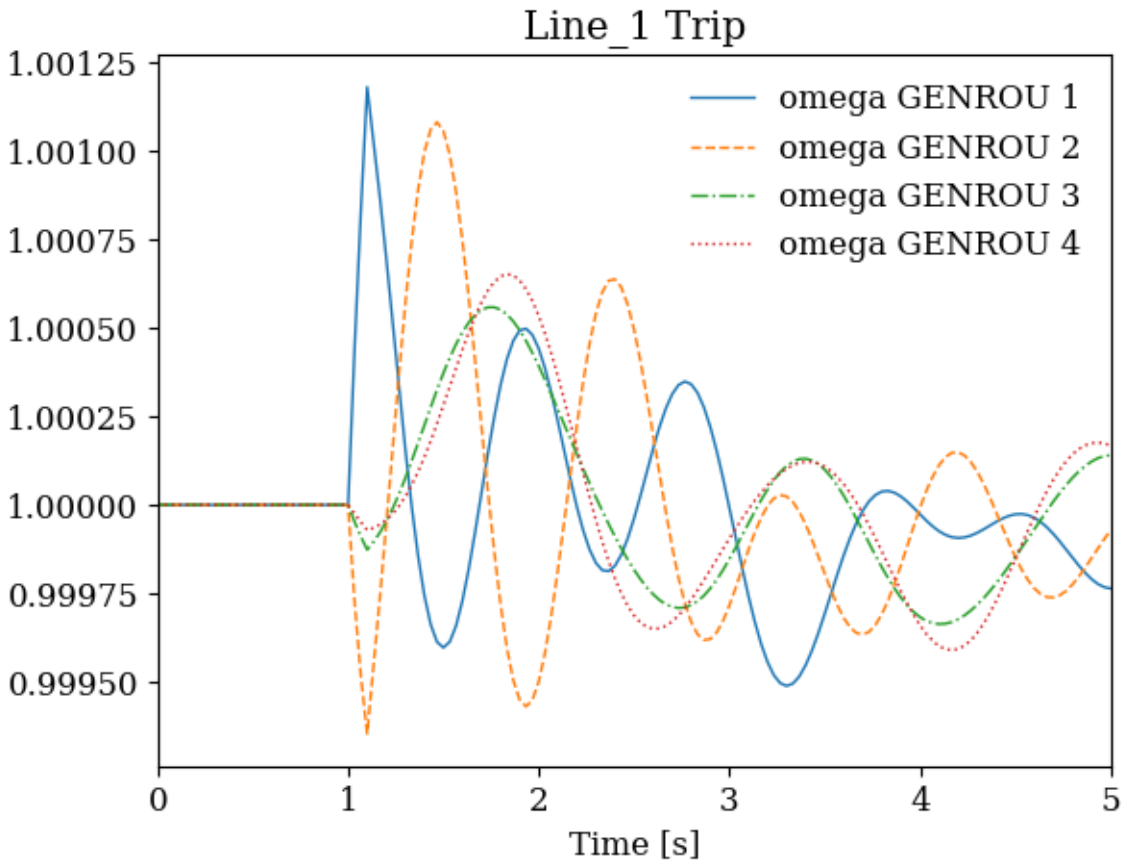
```
# Compare generator response across contingencies
fig, axes = plt.subplots(1, len(results), figsize=(12, 4))

for ax, (line_idx, ss) in zip(axes, results.items()):
    ss.TDS.plt.plot(ss.GENROU.omega, ax=ax,
```

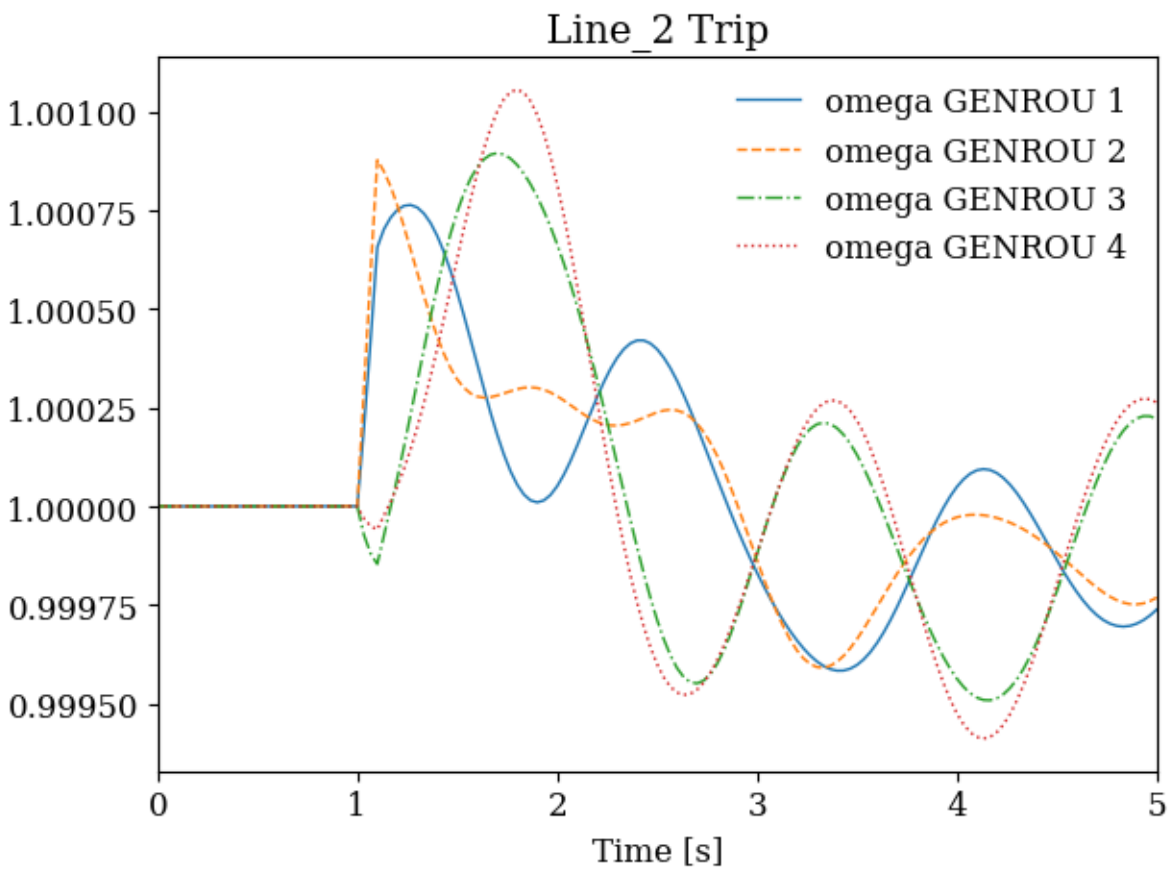
(continues on next page)

(continued from previous page)

```
                    title=f'{line_idx} Trip', latex=False, show=False)

plt.tight_layout()
plt.show()
```




Line_1 Trip
1.00125
1.00100
1.00075
1.00050
1.00025
1.00000
0.99975
0.99950
0
1
2
3
4
5
Time [s]
omega GENROU 1
omega GENROU 2
omega GENROU 3
omega GENROU 4



2.9.4 Choosing the Right Approach

The following table summarizes when to use each batch processing technique:

Approach	Best For	Parallelism	Memory Usage
File-based CLI	Large studies (>100 cases), production runs	Multi-process	Low (one case at a time)
`pool=True` API	Medium studies, need results in memory	Multi-process	High (all results in memory)
Python loop	Small studies, rapid prototyping	Single-thread	Controlled (can discard results)

For studies with more than about 10 cases, the file-based parallel approach is usually fastest because it fully utilizes all CPU cores. For smaller studies or when you are actively developing and testing, the simpler Python loop avoids the overhead of file I/O.

2.9.5 Cleanup

```
shutil.rmtree('batch_cases', ignore_errors=True)
!andes misc -C
```

```
"/home/docs/checkouts/readthedocs.org/user_builds/andes/checkouts/stable/docs/
↪source/tutorials/kundur_p_12.75_out.lst" removed.
"/home/docs/checkouts/readthedocs.org/user_builds/andes/checkouts/stable/docs/
↪source/tutorials/kundur_full_out.txt" removed.
"/home/docs/checkouts/readthedocs.org/user_builds/andes/checkouts/stable/docs/
↪source/tutorials/kundur_p_11.59_out.lst" removed.
"/home/docs/checkouts/readthedocs.org/user_builds/andes/checkouts/stable/docs/
↪source/tutorials/kundur_p_12.75_out.txt" removed.
"/home/docs/checkouts/readthedocs.org/user_builds/andes/checkouts/stable/docs/
↪source/tutorials/kundur_full_out.lst" removed.
"/home/docs/checkouts/readthedocs.org/user_builds/andes/checkouts/stable/docs/
↪source/tutorials/kundur_p_12.75_out.npz" removed.
"/home/docs/checkouts/readthedocs.org/user_builds/andes/checkouts/stable/docs/
↪source/tutorials/kundur_p_11.59_out.npz" removed.
"/home/docs/checkouts/readthedocs.org/user_builds/andes/checkouts/stable/docs/
↪source/tutorials/kundur_p_13.91_out.lst" removed.
"/home/docs/checkouts/readthedocs.org/user_builds/andes/checkouts/stable/docs/
↪source/tutorials/kundur_full_out.npz" removed.
"/home/docs/checkouts/readthedocs.org/user_builds/andes/checkouts/stable/docs/
↪source/tutorials/kundur_p_13.91_out.npz" removed.
"/home/docs/checkouts/readthedocs.org/user_builds/andes/checkouts/stable/docs/
↪source/tutorials/kundur_p_13.91_out.txt" removed.
"/home/docs/checkouts/readthedocs.org/user_builds/andes/checkouts/stable/docs/
↪source/tutorials/kundur_p_11.59_out.txt" removed.
```

2.9.6 Next Steps

- *Contingency Analysis* - Systematic N-1 contingency screening
- *Dynamic Control and Setpoint Changes* - Runtime parameter modifications
- *Frequency Response and Load Shedding* - Frequency response analysis

2.10 Contingency Analysis

Contingency analysis is a critical component of power system planning and operations. It evaluates how the system responds to various disturbances, typically single component outages known as "N-1" contingencies. By systematically simulating line trips, generator outages, and bus faults, you can identify weak points in the system that might lead to voltage collapse or loss of synchronism.

This tutorial demonstrates how to perform systematic contingency screening using ANDES. We cover N-1 line contingencies, generator trips, fault-and-clear analysis, and methods for assessing stability from simulation results.

Note: **Prerequisites:** This tutorial uses Python functions and pandas DataFrames. Complete *Time-Domain Simulation* for disturbance basics and *Parameter Sweeps and Batch Processing* for batch simulation patterns before proceeding.

2.10.1 Setup

```
%matplotlib inline

import andes
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt

andes.config_logger(stream_level=30)  # Reduce verbosity for batch runs
```

```
case_file = andes.get_case('kundur/kundur_full.xlsx')
```

2.10.2 N-1 Line Contingency Screening

The most common contingency analysis involves testing the loss of each transmission line individually. For each line, we load a fresh system, add a Toggle device to trip the line at a specified time, run the simulation, and then assess whether the system remained stable.

First, let us identify all available lines in the test case.

```
# Load system to inspect lines
ss = andes.load(case_file, setup=False)
all_lines = list(ss.Line.idx.v)
print(f"Lines in the system: {all_lines}")
```

```
Lines in the system: ['Line_0', 'Line_1', 'Line_2', 'Line_3', 'Line_4', 'Line_
↪5', 'Line_6', 'Line_7', 'Line_8', 'Line_9', 'Line_10', 'Line_11', 'Line_12',
↪ 'Line_13', 'Line_14']
```

Now we loop through each line and simulate its outage. For demonstration purposes, we test only the first three lines to keep execution time reasonable. In a real study, you would test all lines.

```
# Select subset for demonstration
lines_to_test = all_lines[:3]

results = {}
for line_idx in lines_to_test:
    # Load fresh system with setup=False to allow adding devices
    ss = andes.load(case_file, setup=False)

    # Add line trip at t=1.0s
```

(continues on next page)

(continued from previous page)

```
    ss.add('Toggle', model='Line', dev=line_idx, t=1.0)

    ss.setup()

    # Disable existing Toggle in base case
    ss.Toggle.set_status(1, 0)

    ss.PFlow.run()
    ss.TDS.config.tf = 5
    ss.TDS.config.no_tqdm = 1
    ss.TDS.run()

    # Record max generator speed deviation as a stability metric
    omega = ss.dae.ts.x[:, ss.GENROU.omega.a]
    omega_max = omega.max()
    omega_min = omega.min()

    results[line_idx] = {
        'omega_max': omega_max,
        'omega_min': omega_min,
        'exit_code': ss.exit_code,
        'system': ss
    }

    print(f"{line_idx}: omega_max={omega_max:.4f}, omega_min={omega_min:.4f}")
```

```
Line_0: omega_max=1.0050, omega_min=0.9988
Line_1: omega_max=1.0050, omega_min=0.9988
Line_2: omega_max=1.0085, omega_min=0.9998
```

2.10.3 Stability Assessment

After running simulations, we need to determine which contingencies resulted in stable operation and which caused problems. Common stability criteria include:

- Generator speeds should remain within acceptable bounds (typically 0.95-1.05 pu)
- Bus voltages should stay above minimum thresholds (typically 0.8 pu)
- The simulation should complete without numerical failures

The following function encapsulates these checks.

Tip: The screening loop above and the function below use direct DAE array indexing (`ss.dae.ts.x[:, addr]`) for speed in batch loops. For interactive analysis or single simulations, `ss.TDS.get_timeseries(ss.GENROU.omega)` returns the same data as a pandas DataFrame — see *Time-Domain Simulation* for details.

```
def assess_stability(ss):
    """Return stability assessment for a completed simulation."""

    if ss.exit_code != 0:
        return {'stable': False, 'reason': 'simulation_failed'}

    omega = ss.dae.ts.x[:, ss.GENROU.omega.a]
    v = ss.dae.ts.y[:, ss.Bus.v.a]

    omega_max = omega.max()
    omega_min = omega.min()
    v_min = v.min()

    stable = (omega_max < 1.05 and omega_min > 0.95 and v_min > 0.8)

    return {
        'stable': stable,
        'omega_max': omega_max,
        'omega_min': omega_min,
        'v_min': v_min
    }
```

```
# Generate contingency report
records = []
for cont_id, data in results.items():
    metrics = assess_stability(data['system'])
    records.append({
        'contingency': cont_id,
        **metrics
    })

df = pd.DataFrame(records)
df
```

```
  contingency  stable  omega_max  omega_min     v_min
0      Line_0    True   1.005020   0.998823  0.880968
1      Line_1    True   1.005018   0.998824  0.881003
2      Line_2    True   1.008547   0.999849  0.830137
```

2.10.4 Visualizing Contingency Results

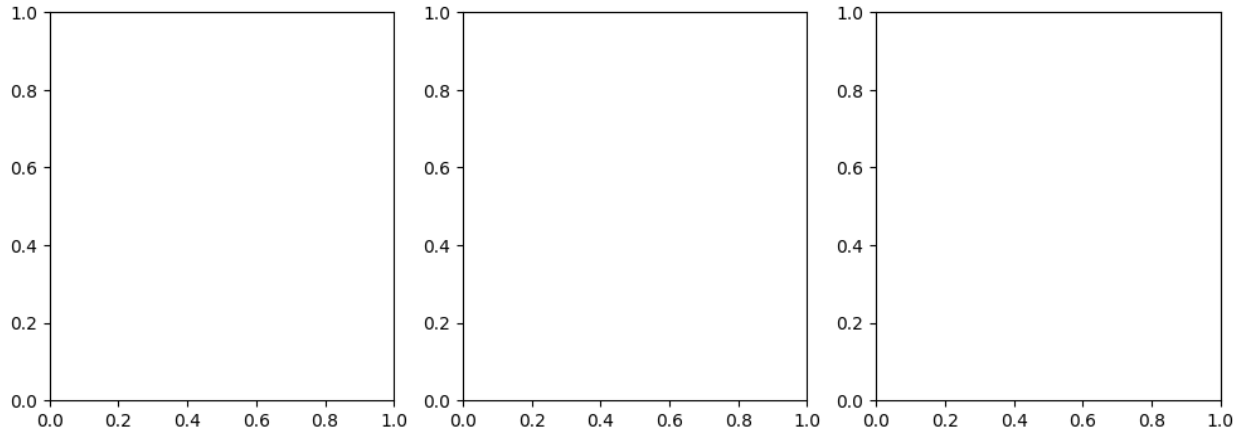
Plotting the generator response for each contingency provides insight into the system dynamics and helps identify which modes are excited by different disturbances.

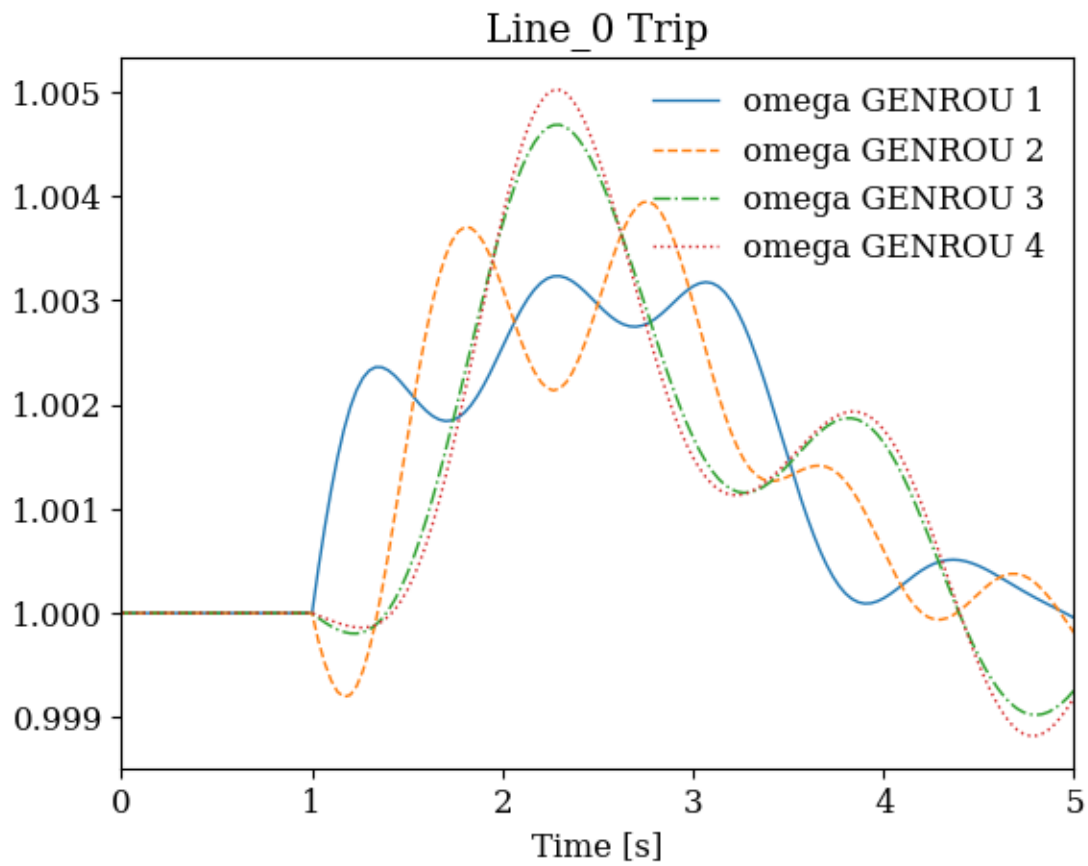
```
fig, axes = plt.subplots(1, len(results), figsize=(12, 4))

for ax, (line_idx, data) in zip(axes, results.items()):
    ss = data['system']
    ss.TDS.plt.plot(ss.GENROU.omega, ax=ax,
                    title=f'{line_idx} Trip', latex=False, show=False)
```

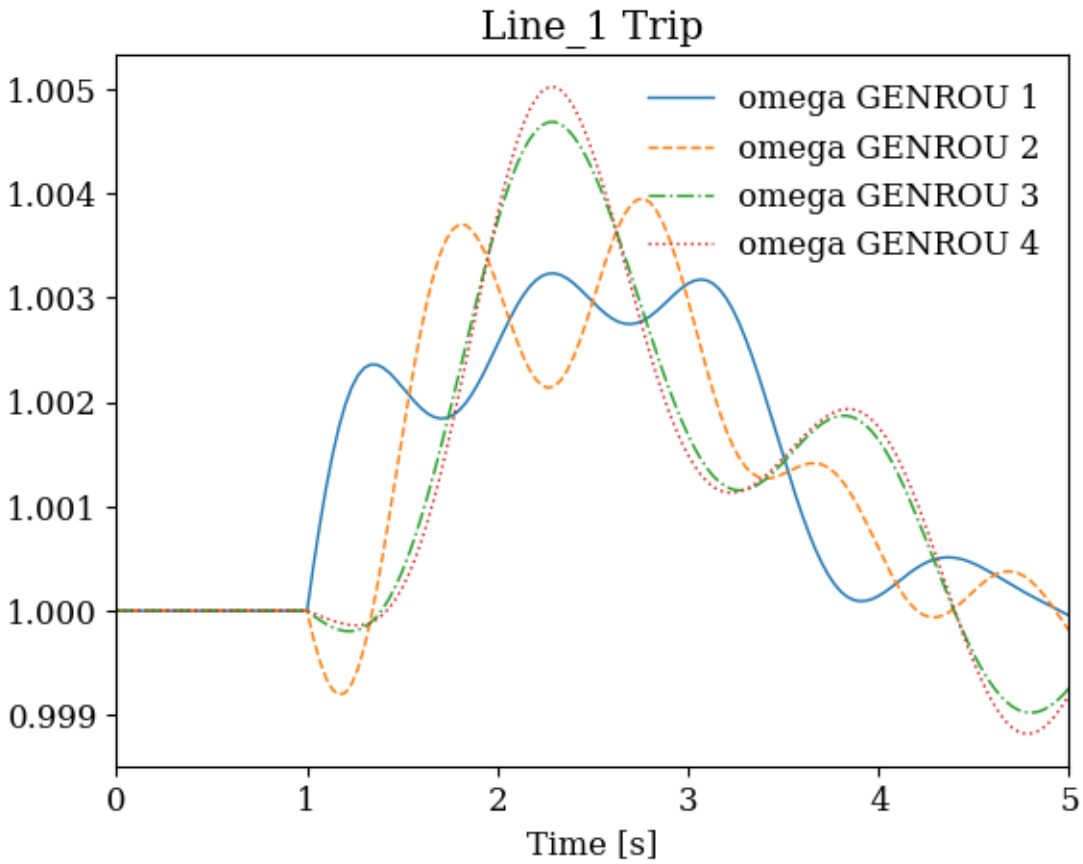
(continues on next page)

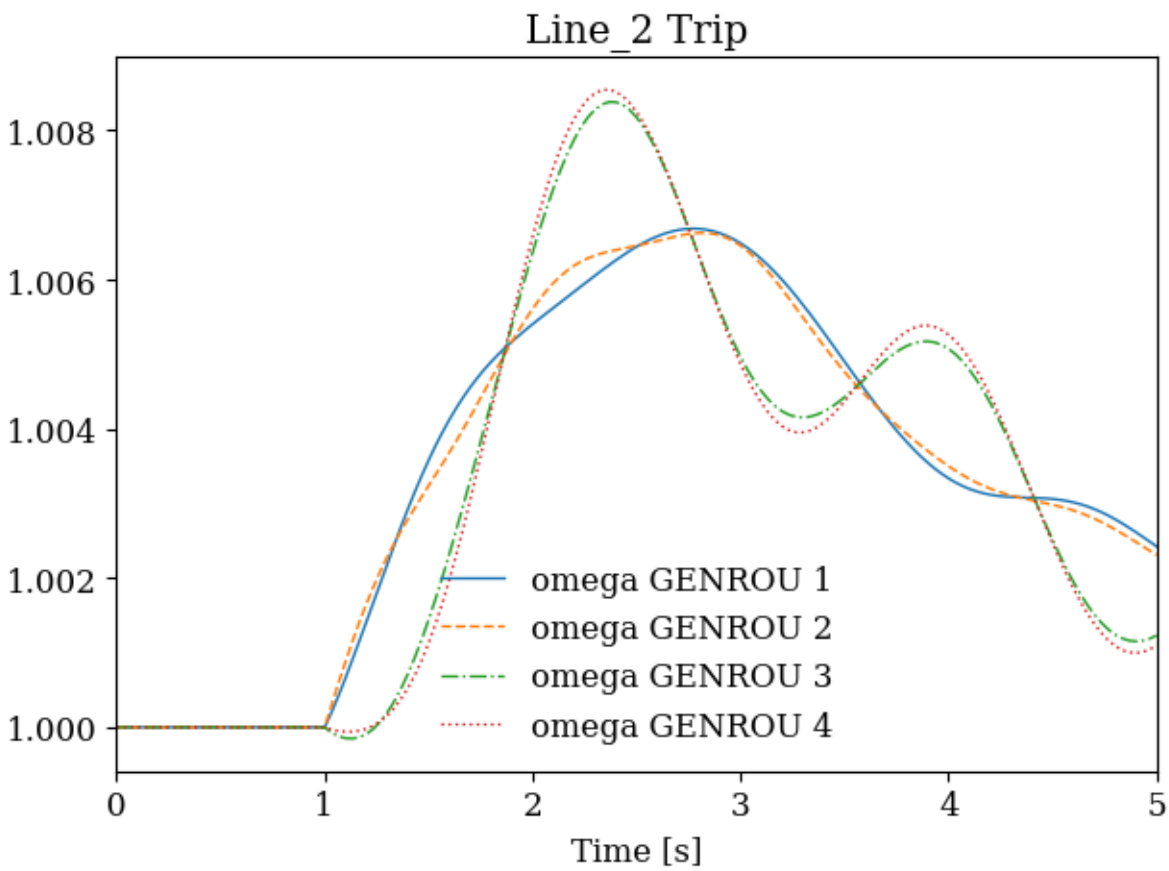
(continued from previous page)

```
plt.tight_layout()
plt.show()
```






Line_1 Trip
omega GENROU 1
omega GENROU 2
omega GENROU 3
omega GENROU 4
1.005
1.004
1.003
1.002
1.001
1.000
0.999
0
1
2
3
4
5
Time [s]



2.10.5 Fault-and-Clear Analysis

Three-phase faults followed by fault clearing represent more severe disturbances than simple line trips. The severity depends on both the fault location (which bus) and the fault duration (clearing time). Faults near generators or at critical buses can cause rapid acceleration of nearby machines.

In ANDES, faults are modeled using the `Fault` device with `tf` (fault time) and `tc` (clearing time) parameters.

```
# Get bus indices for fault analysis
ss = andes.load(case_file, setup=False)
print(f"Available buses: {ss.Bus.idx.v}")
```

```
Available buses: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

```
# Test 100ms fault at bus 3
ss = andes.load(case_file, setup=False)

# Add fault at t=1.0s, clear at t=1.1s (100ms duration)
ss.add('Fault', bus=3, tf=1.0, tc=1.1)

ss.setup()
```

(continues on next page)

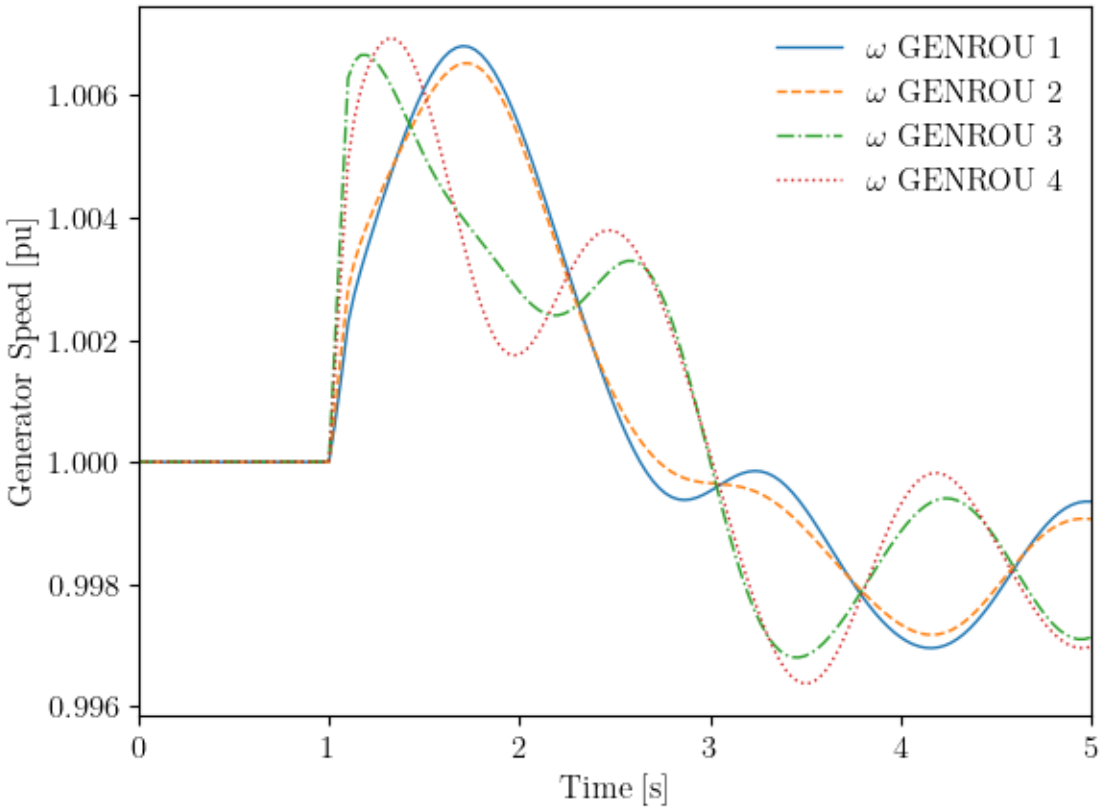
(continued from previous page)

```
# Disable existing Toggle in base case
ss.Toggle.set_status(1, 0)

ss.PFlow.run()
ss.TDS.config.tf = 5
ss.TDS.run()
```

```
  0%|          | 0/100 [00:00<?, ?%/s]
```

```
True
```

```
# Plot response to fault
fig, ax = ss.TDS.plt.plot(ss.GENROU.omega, ylabel='Generator Speed [pu]')
```



2.10.6 Critical Clearing Time

Critical Clearing Time (CCT) is the maximum fault duration for which the system remains stable. It is a key metric for protection system coordination. Finding CCT involves a binary search: if the system is stable at a given clearing time, we try a longer fault; if unstable, we try a shorter fault.

The following function implements this binary search.

```
def find_cct(case_file, bus_idx, t_fault=1.0,
             tc_min=0.05, tc_max=0.5, tol=0.02):
    """Binary search for critical clearing time.

    Parameters
    ----------
    case_file : str
        Path to the case file
    bus_idx : int or str
        Bus index for the fault
    t_fault : float
        Fault initiation time
    tc_min : float
        Minimum clearing time to test
    tc_max : float
        Maximum clearing time to test
    tol : float
        Tolerance for binary search (seconds)

    Returns
    -------
    float
        Estimated critical clearing time
    """
    iterations = 0
    max_iterations = 20  # Safety limit

    while (tc_max - tc_min) > tol and iterations < max_iterations:
        tc_mid = (tc_min + tc_max) / 2

        ss = andes.load(case_file, setup=False)
        ss.add('Fault', bus=bus_idx, tf=t_fault, tc=t_fault + tc_mid)
        ss.setup()

        # Disable existing Toggle
        ss.Toggle.set_status(1, 0)

        ss.PFlow.run()
        ss.TDS.config.tf = 5
        ss.TDS.config.no_tqdm = 1
        ss.TDS.run()

        metrics = assess_stability(ss)

        if metrics['stable']:
```

(continues on next page)

(continued from previous page)

```
            tc_min = tc_mid  # Stable, try longer fault
        else:
            tc_max = tc_mid  # Unstable, try shorter fault

        iterations += 1

    return tc_min
```

```
# Find CCT for bus 3
cct = find_cct(case_file, bus_idx=3)
print(f"Critical Clearing Time for Bus 3: {cct*1000:.0f} ms")
```

```
Critical Clearing Time for Bus 3: 50 ms
```

2.10.7 Stability Metrics Summary

The following table summarizes common metrics used in contingency screening:

Metric	Description	Typical Threshold
`omega_max`	Maximum generator speed	< 1.05 pu
`omega_min`	Minimum generator speed	> 0.95 pu
`v_min`	Minimum bus voltage	> 0.8 pu
`exit_code`	Simulation completion	== 0
CCT	Critical clearing time	Depends on protection

These thresholds should be adjusted based on the specific system and operating standards applicable to your study.

2.10.8 Cleanup

```
!andes misc -C
```

```
"/home/docs/checkouts/readthedocs.org/user_builds/andes/checkouts/stable/docs/
↪source/tutorials/kundur_full_out.txt" removed.
"/home/docs/checkouts/readthedocs.org/user_builds/andes/checkouts/stable/docs/
↪source/tutorials/kundur_full_out.lst" removed.
"/home/docs/checkouts/readthedocs.org/user_builds/andes/checkouts/stable/docs/
↪source/tutorials/kundur_full_out.npz" removed.
```

2.10.9 Next Steps

- *Dynamic Control and Setpoint Changes* - Runtime parameter modifications
- *Frequency Response and Load Shedding* - Frequency response analysis
- *Parameter Sweeps and Batch Processing* - Batch simulation techniques

2.11 Dynamic Control and Setpoint Changes

Time-domain simulations typically use fixed parameters throughout the simulation period. However, many advanced applications require changing control setpoints during simulation, such as implementing economic dispatch schedules, testing automatic generation control (AGC), or integrating with reinforcement learning algorithms. ANDES supports this through its multi-stage simulation capability.

The key insight is that `ss.TDS.run()` can be called multiple times, each time advancing the simulation from the current state to a new end time. Between calls, you can modify setpoints, reference values, or other parameters, and the simulation will continue from exactly where it left off.

This tutorial demonstrates how to change generator power setpoints during simulation and observe the dynamic response.

Note: **Prerequisites:** Complete *Time-Domain Simulation* for time-domain simulation basics and *Data and File Formats* for parameter modification techniques.

2.11.1 Setup

```
%matplotlib inline

import andes

andes.config_logger(stream_level=20)
```

2.11.2 Initialize the System

First, we load the system and run power flow to establish the initial operating point. We also disable any pre-existing disturbances (like the Toggle in the Kundur test case) so we can observe only the effects of our setpoint changes.

```
kundur = andes.get_case('kundur/kundur_full.xlsx')
ss = andes.run(kundur)
```

```
Working directory: "/home/docs/checkouts/readthedocs.org/user_builds/andes/
↪checkouts/stable/docs/source/tutorials"
> Loaded generated Python code in "/home/docs/.andes/pycode".
```

(continues on next page)

(continued from previous page)

```
Parsing input file "/home/docs/checkouts/readthedocs.org/user_builds/andes/
↪envs/stable/lib/python3.11/site-packages/andes/cases/kundur/kundur_full.xlsx
↪"...
Input file parsed in 0.0306 seconds.
Connectivity check completed in 0.0001 seconds.
-> System connectivity check results:
  No islanded bus detected.
  System is interconnected.
  Each island has a slack bus correctly defined and enabled.
System internal structure set up in 0.0200 seconds.

-> Power flow calculation
           Numba: Off
   Sparse solver: KLU
 Solution method: NR method
Power flow initialized in 0.0031 seconds.
0: |F(x)| = 14.9282832
1: |F(x)| = 3.608627841
2: |F(x)| = 0.1701107882
3: |F(x)| = 0.002038626956
4: |F(x)| = 3.745103977e-07
Converged in 5 iterations in 0.0025 seconds.
Report saved to "kundur_full_out.txt" in 0.0008 seconds.
-> Single process finished in 0.6020 seconds.
```

```
# Disable the pre-existing Toggle to have a clean experiment
ss.Toggle.set_status(1, 0)
```

2.11.3 Understanding Turbine Governor Setpoints

ANDES provides a group-level setpoint API that automatically resolves the controller chain. For example, `ss.SynGen.set_paux()` locates the turbine governor connected to a given generator and writes to its auxiliary power input. The user does not need to know which governor model is in use.

Under the hood, the TGOV1 turbine governor model contains a power reference input `pref0` and an auxiliary power input `paux0`. The following code examines the TGOV1 model structure to illustrate where these setpoints enter the equations.

```
# View the TGOV1 algebraic equations to see how pref0 and paux0 are used
print("TGOV1 Algebraic Equations:")
for var_name in ['pref', 'paux', 'pd']:
    var = getattr(ss.TGOV1, var_name)
    print(f"  {var_name}: {var.e_str}")
```

```
TGOV1 Algebraic Equations:
  pref: pref0 * R - pref
  paux: paux0 - paux
  pd: ue*(- wd + pref + paux) * gain - pd
```

The equation `pref0 - pref = 0` indicates that the algebraic variable `pref` tracks the parameter `pref0`, which serves as the base power reference. Similarly, `paux0 - paux = 0` tracks the auxiliary power input `paux0`. Both `pref` and `paux` feed into the `pd` equation, ultimately affecting the turbine mechanical power output.

The group-level API `set_paux()` writes to `paux0`, while `set_pref()` writes to `pref0`. Both methods resolve the controller chain automatically.

The initial values of `paux0` for all generators are shown below.

```
print(f"Initial paux0 values: {ss.TGOV1.paux0.v}")
```

```
Initial paux0 values: [0. 0. 0. 0.]
```

2.11.4 Stage 1: Run Simulation to Steady State

We first run the simulation for one second with no setpoint changes. This establishes a baseline and confirms the system is in steady state before we apply any changes.

```
# Set the first stopping time
ss.TDS.config.tf = 1
ss.TDS.run()
```

```
DAE compaction: removed 5 algebraic variable slots (m: 149 -> 144)
Initialization for dynamics completed in 0.0166 seconds.
Initialization was successful.
Simulation to t=1.00 sec completed in 0.0316 seconds.
Outputs to "kundur_full_out.lst" and "kundur_full_out.npz".
Outputs written in 0.0017 seconds.
```

```
  0%|          | 0/100 [00:00<?, ?%/s]
```

```
True
```

Finding a Generator by Bus Number

A common task is "change the voltage setpoint of the generator at bus 2" — but you only know the bus number, not the generator index. The `find_connected()` method bridges this gap by returning all devices that reference a given bus or device.

Combined with the setpoint API, this eliminates the need to manually trace through PV → StaticGen → GENROU → exciter → VREF0.

```
# Step 1: Find what's connected to bus 2
connected = ss.find_connected('Bus', 2)
print("Devices at bus 2:")
for model, idxes in connected.items():
    print(f"  {model}: {idxes}")
```

(continues on next page)

(continued from previous page)

```
# Step 2: Get the generator idx from the result
gen_at_bus2 = connected['GENROU'][0]
print(f"\nGenerator at bus 2: {gen_at_bus2}")

# Step 3: Read and change the voltage setpoint
original_vref = ss.SynGen.get_vref(ss, gen_at_bus2)
print(f"Current vref: {original_vref:.4f}")

ss.SynGen.set_vref(ss, gen_at_bus2, 1.05)
print(f"Updated vref: {ss.SynGen.get_vref(ss, gen_at_bus2):.4f}")

# Restore for the rest of this tutorial (which demonstrates set_paux)
ss.SynGen.set_vref(ss, gen_at_bus2, original_vref)
```

```
Devices at bus 2:
  PV: [2]
  Line: ['Line_12']
  GENROU: [2]

Generator at bus 2: 2
Current vref: 1.1010
Updated vref: 1.0500
```

2.11.5 Stage 2: Apply Setpoint Change

The auxiliary power setpoint of the first generator is increased by 0.05 pu. This change simulates a dispatch command instructing the generator to increase its output.

The recommended approach is the group-level setpoint API, which resolves the controller chain automatically. The generator index is specified, and the API locates the connected governor internally.

```
# Get the first generator's idx
gen_idx = ss.GENROU.idx.v[0]

# Increase the auxiliary power by 0.05 pu (system base)
ss.SynGen.set_paux(ss, gen_idx, 0.05)

print(f"Updated paux: {ss.SynGen.get_paux(ss, gen_idx)}")
```

```
Updated paux: 0.05
```

```
# Continue simulation to 2 seconds
ss.TDS.config.tf = 2
ss.TDS.run()
```

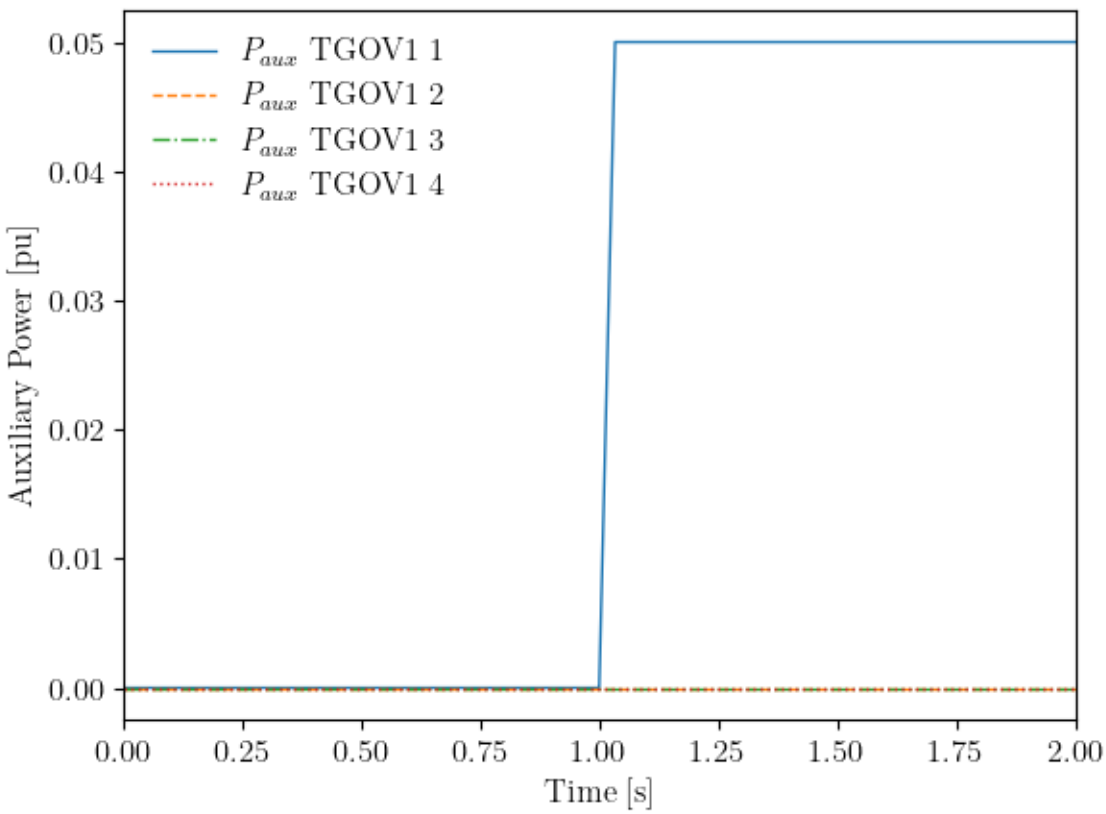
```
  0%|          | 0/100 [00:00<?, ?%/s]
```

```
Simulation to t=2.00 sec completed in 0.0638 seconds.
Outputs to "kundur_full_out.lst" and "kundur_full_out.npz".
Outputs written in 0.0035 seconds.
```

```
True
```

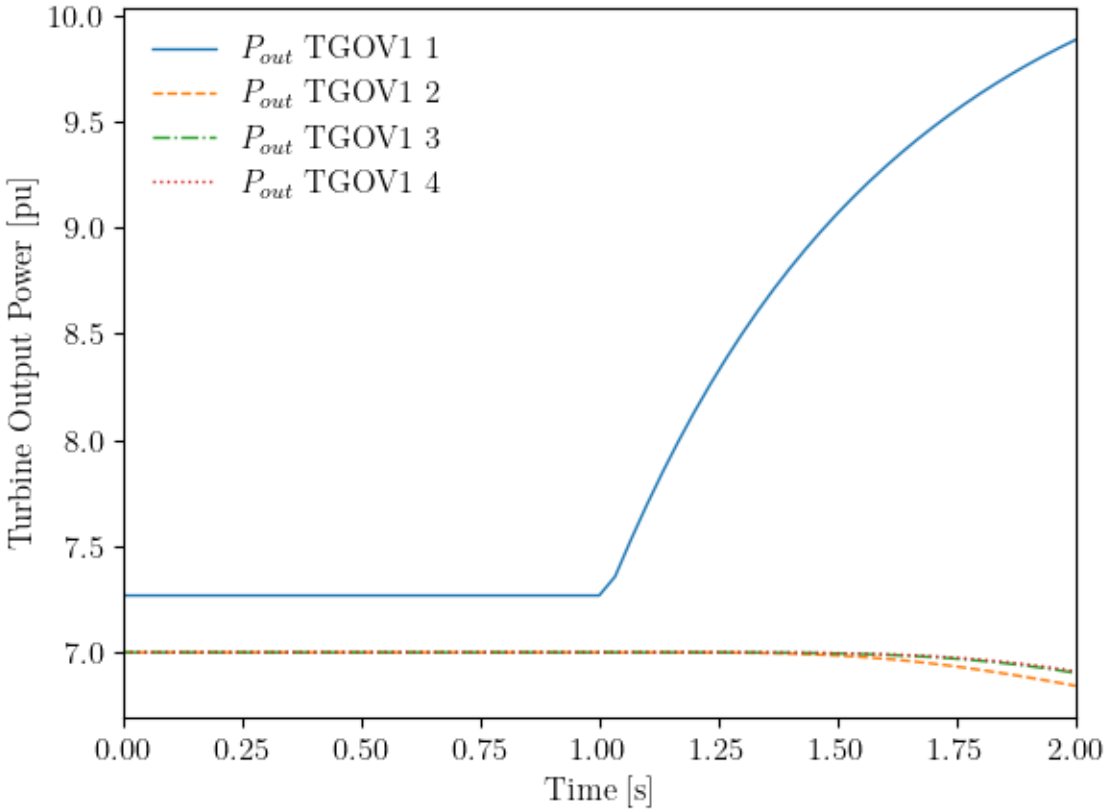
Let us plot the auxiliary power input to see the step change.

```
fig, ax = ss.TDS.plt.plot(ss.TGOV1.paux, ylabel='Auxiliary Power [pu]')
```



The plot shows that `paux` for the first generator stepped from 0 to 0.05 pu at t=1 second. This change propagates through the turbine governor dynamics to affect the mechanical power output.

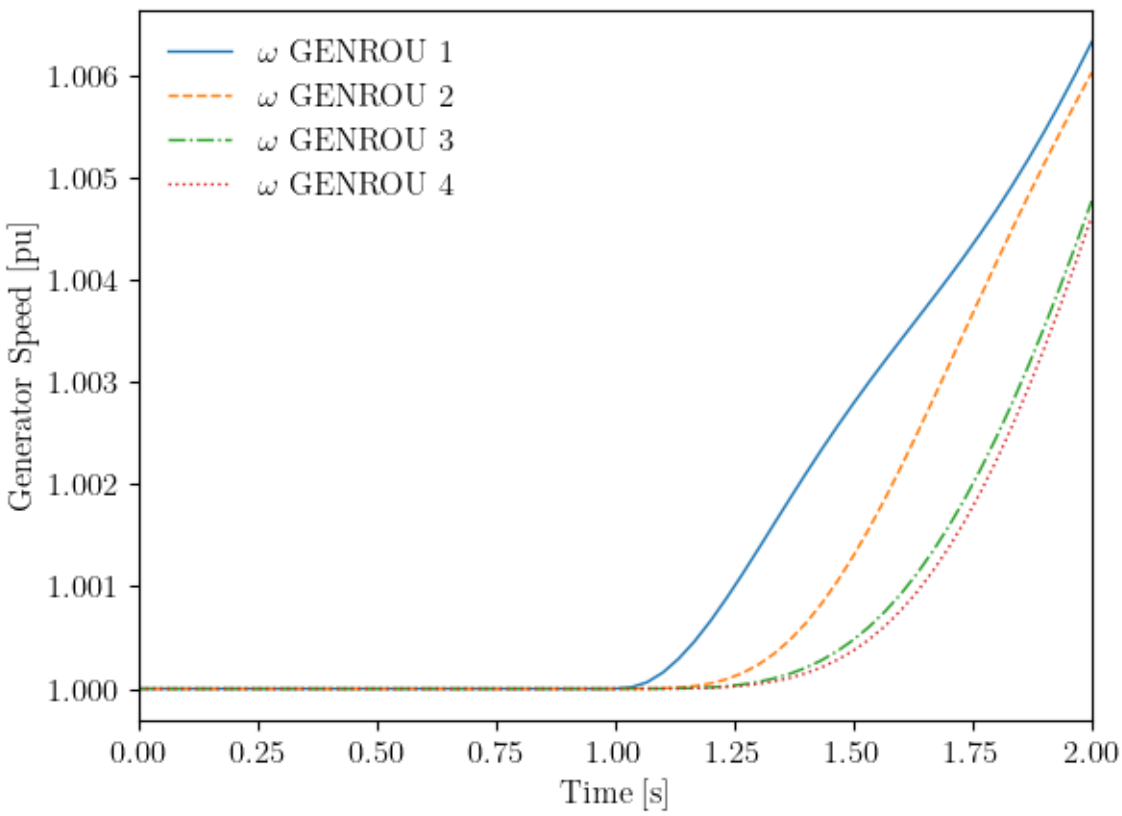
```
fig, ax = ss.TDS.plt.plot(ss.TGOV1.pout, ylabel='Turbine Output Power [pu]')
```



The turbine output power increases for the first generator (blue curve) following the setpoint change. The response is not instantaneous because the turbine governor has dynamics (lag and lead-lag blocks) that smooth the response.

The generator speeds also respond to this power imbalance.

```
fig, ax = ss.TDS.plt.plot(ss.GENROU.omega, ylabel='Generator Speed [pu]')
```



2.11.6 Stage 3: Restore Setpoint

The auxiliary power is reset to zero, and the system is observed returning to its original operating point. This demonstrates that the simulation correctly tracks multiple setpoint changes.

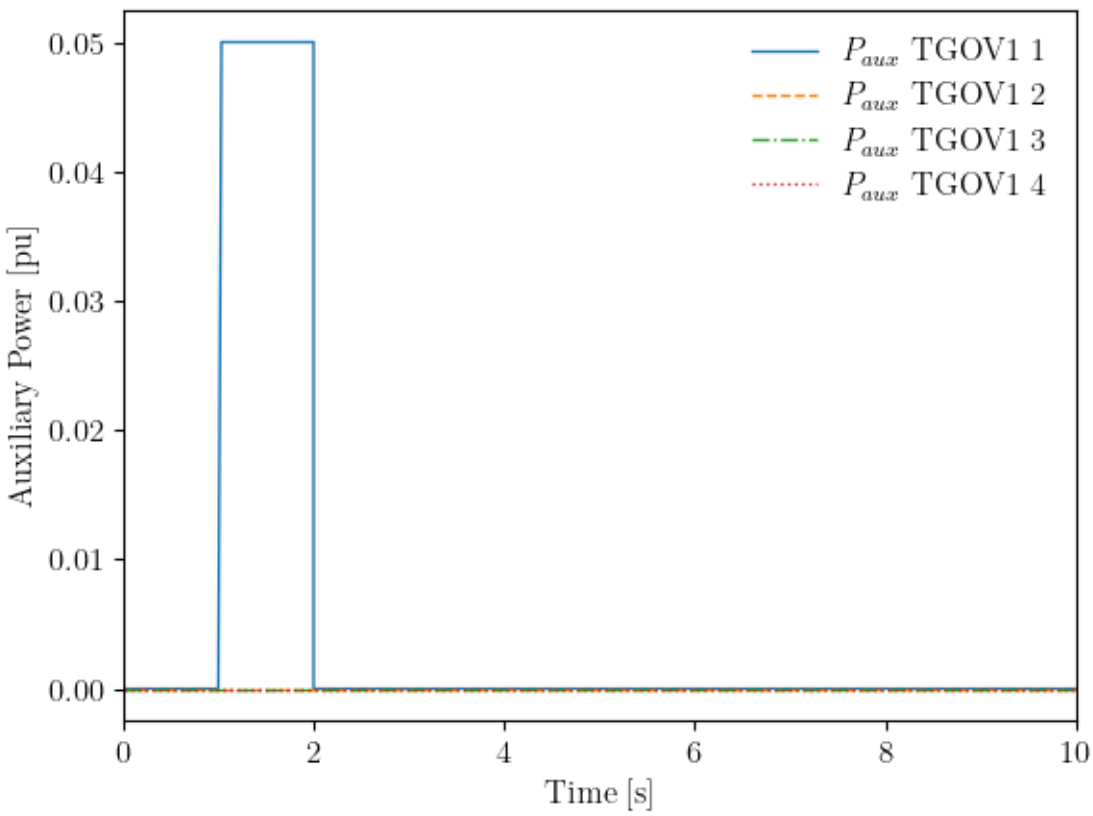
```
# Reset the auxiliary power
ss.SynGen.set_paux(ss, gen_idx, 0.0)

# Continue simulation to 10 seconds
ss.TDS.config.tf = 10
ss.TDS.run()
```

```
  0%|          | 0/100 [00:00<?, ?%/s]
```

```
Simulation to t=10.00 sec completed in 0.4967 seconds.
Outputs to "kundur_full_out.lst" and "kundur_full_out.npz".
Outputs written in 0.0139 seconds.
```

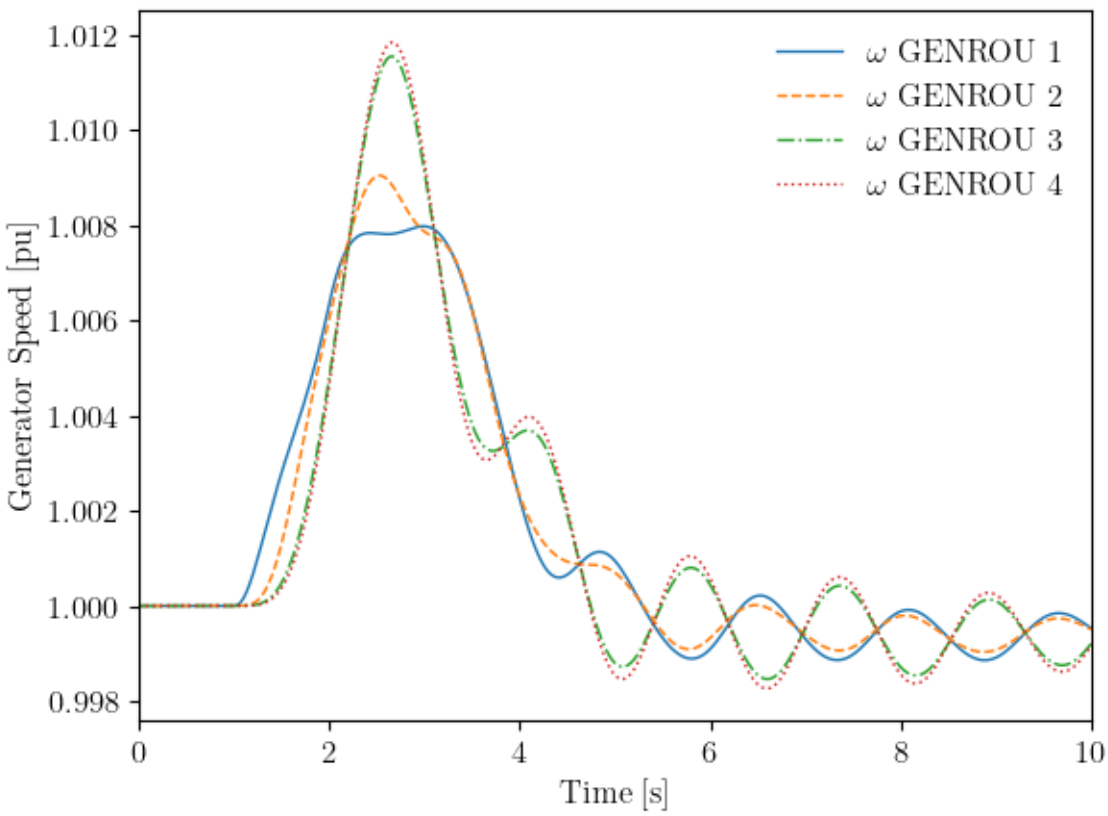
```
True
```

```
fig, ax = ss.TDS.plt.plot(ss.TGOV1.paux, ylabel='Auxiliary Power [pu]')
```



The auxiliary power shows two transitions: up at t=1s and back down at t=2s. The generator speeds show the corresponding dynamics as the system adjusts to each change.

```
fig, ax = ss.TDS.plt.plot(ss.GENROU.omega, ylabel='Generator Speed [pu]')
```



2.11.7 Applications

The multi-stage simulation capability enables several advanced applications. The group-level setpoint API (`set_pref`, `set_vref`, `set_paux`, `set_qref`) is recommended because it resolves the controller chain automatically.

Application	Recommended API	Description
Economic dispatch	`SynGen.set_pref()`	Follow dispatch schedules
AGC simulation	`SynGen.set_paux()`	Frequency regulation signals
Voltage control	`SynGen.set_vref()`	Voltage setpoint tracking
Renewable dispatch	`RenGen.set_pref()` / `set_qref()`	P and Q reference changes
Reinforcement learning	`GroupBase.set_setpoint()`	Generic setpoint interface

The key technique is the same in all cases: the simulation is run to a checkpoint, the relevant setpoint is modified through the API, and the simulation is continued.

Note: Advanced usage. For bulk operations or model-specific parameters not covered by the setpoint API (e.g., modifying all governor `paux0` values simultaneously), direct in-place array assignment remains available:

```
ss.TGOV1.paux0.v[:] = new_values  # in-place update of the entire array
```

When modifying arrays directly, in-place assignment (e.g., `array[0] = value` or `array[:] = values`) must be used rather than replacing the array reference. Replacing the reference (e.g., `array = new_array`) disconnects the variable from the simulation.

2.11.8 Cleanup

```
!andes misc -C
```

```
"/home/docs/checkouts/readthedocs.org/user_builds/andes/checkouts/stable/docs/
↪source/tutorials/kundur_full_out.txt" removed.
"/home/docs/checkouts/readthedocs.org/user_builds/andes/checkouts/stable/docs/
↪source/tutorials/kundur_full_out.lst" removed.
"/home/docs/checkouts/readthedocs.org/user_builds/andes/checkouts/stable/docs/
↪source/tutorials/kundur_full_out.npz" removed.
```

2.11.9 Next Steps

- *Frequency Response and Load Shedding* - Frequency response analysis
- *Eigenvalue Analysis* - Small-signal stability analysis
- *Parameter Sweeps and Batch Processing* - Batch simulation techniques

2.12 Frequency Response and Load Shedding

System frequency is a direct indicator of the real power balance between generation and load. When a generator trips unexpectedly, the remaining generators must pick up the lost generation through their turbine governors. If the generation shortfall is too large, frequency will decline until it stabilizes at a new (lower) value determined by the governor droop characteristics, or protective relays will disconnect loads to restore balance.

This tutorial demonstrates how to study frequency response in ANDES by simulating a generator trip and then implementing load shedding to restore frequency. These techniques are essential for understanding power system resilience and designing emergency control schemes.

Note: **Prerequisites:** Complete *Time-Domain Simulation* for disturbance simulation and *Dynamic Control and Setpoint Changes* for multi-stage simulation techniques.

2.12.1 Setup

```
%matplotlib inline

import andes
import numpy as np

andes.config_logger(stream_level=20)
```

2.12.2 Simulating a Generator Trip

We use the IEEE 14-bus system, which includes generators with turbine governors and exciters. The `Toggle` device can disconnect any component, including generators. We will trip one of the generators and observe the frequency response of the remaining machines.

First, load the system with `setup=False` so we can add the Toggle before finalizing the system structure.

```
ieee14_raw = andes.get_case("ieee14/ieee14.raw")
ieee14_dyr = andes.get_case("ieee14/ieee14.dyr")

ss = andes.load(ieee14_raw, addfile=ieee14_dyr, setup=False)
```

```
Working directory: "/home/docs/checkouts/readthedocs.org/user_builds/andes/
↪checkouts/stable/docs/source/tutorials"
> Loaded generated Python code in "/home/docs/.andes/pycode".
Parsing input file "/home/docs/checkouts/readthedocs.org/user_builds/andes/
↪envs/stable/lib/python3.11/site-packages/andes/cases/ieee14/ieee14.raw"...
  IEEE 14 BUS TEST CASE
  03/06/14 CONTO           100.0  1962 W
Input file parsed in 0.0596 seconds.
Parsing additional file "/home/docs/checkouts/readthedocs.org/user_builds/
↪andes/envs/stable/lib/python3.11/site-packages/andes/cases/ieee14/ieee14.dyr
↪"...
Addfile parsed in 0.2950 seconds.
```

```
# Add a Toggle to trip GENROU_2 at t=1 second
ss.add("Toggle", model='SynGen', dev="GENROU_2", t=1.0)

# Complete system setup
ss.setup()
```

```
IEEEST <IEEEST_1> added BusFreq <BusFreq_1> linked to bus <3.0>
ST2CUT <ST2CUT_2> added BusFreq <BusFreq_2> linked to bus <1.0>
ST2CUT <ST2CUT_3> added BusFreq <BusFreq_3> linked to bus <2.0>
Connectivity check completed in 0.0001 seconds.
-> System connectivity check results:
  No islanded bus detected.
  System is interconnected.
  Each island has a slack bus correctly defined and enabled.
System internal structure set up in 0.0222 seconds.
```

```
True
```

The IEEE 14-bus test case includes pre-defined Toggle devices for line switching. We disable them to study the generator trip in isolation.

```
# View all Toggle devices
ss.Toggle.as_df()
```

```
          idx    u      name   model       dev    t
uid
0    Toggle_1  1.0  Toggle_1    Line    Line_1  1.0
1    Toggle_2  1.0  Toggle_2    Line    Line_1  1.1
2    Toggle_3  1.0  Toggle_3  SynGen  GENROU_2  1.0
```

```
# Disable the line switching Toggles (indices 0 and 1)
ss.Toggle.u.v[[0, 1]] = 0
```

For this study, we configure the PQ loads to use constant power (rather than constant impedance) behavior during the simulation. This makes the frequency response more pronounced because the loads do not naturally decrease as voltage drops.

```
# Configure PQ loads for constant power behavior
ss.PQ.config.p2p = 1  # Active power: 100% constant P
ss.PQ.config.q2q = 1  # Reactive power: 100% constant Q
ss.PQ.config.p2z = 0  # No constant impedance component
ss.PQ.config.q2z = 0

# Disable under-voltage PQ-to-Z conversion
ss.PQ.pq2z = 0
```

```
# Run power flow and time-domain simulation
ss.PFlow.run()

ss.TDS.config.tf = 20
ss.TDS.config.criteria = 0  # Disable angle separation criterion (for this␣
↪study)
ss.TDS.run()
```

```
-> Power flow calculation
           Numba: Off
   Sparse solver: KLU
 Solution method: NR method
Power flow initialized in 0.0039 seconds.
0: |F(x)| = 0.5605182134
1: |F(x)| = 0.006202200332
2: |F(x)| = 5.819382827e-06
3: |F(x)| = 6.957087684e-12
Converged in 4 iterations in 0.0023 seconds.
Report saved to "ieee14_out.txt" in 0.0007 seconds.
DAE compaction: removed 6 algebraic variable slots (m: 246 -> 240)
```

(continues on next page)

(continued from previous page)

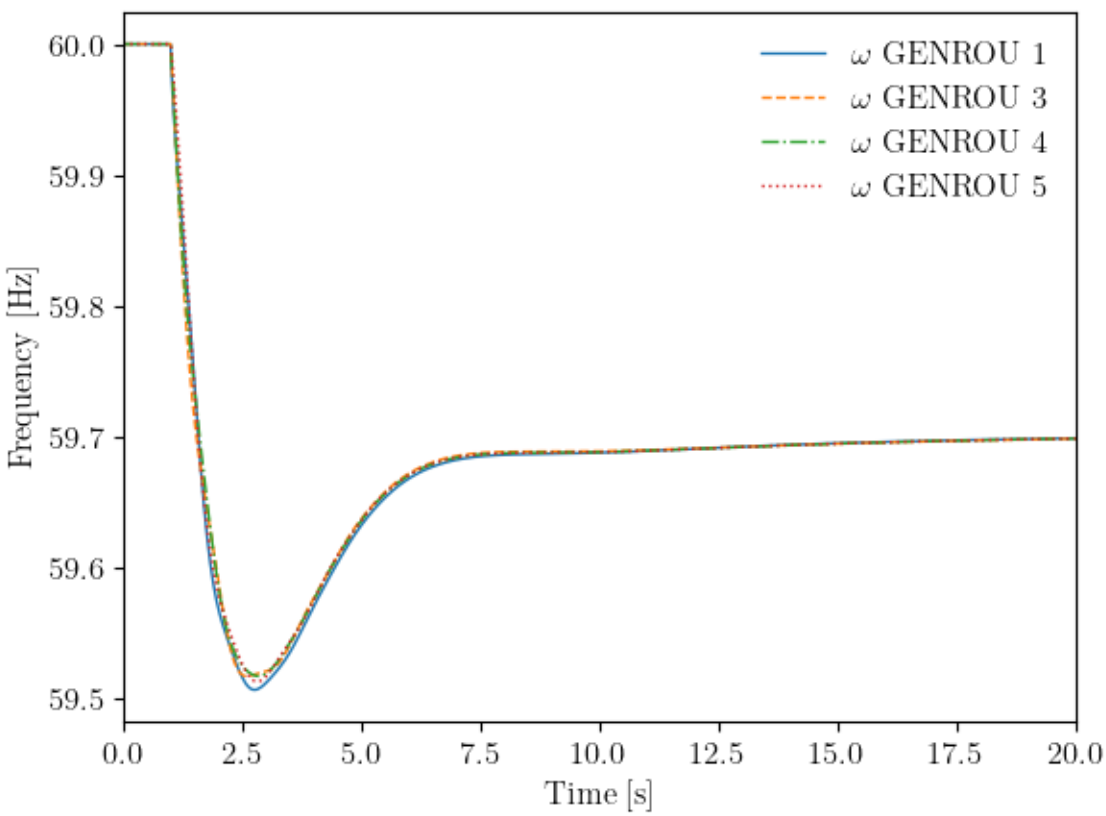
```
Initialization for dynamics completed in 0.0261 seconds.
Initialization was successful.
<Toggle Toggle_3>: SynGen.GENROU_2 status changed to 0 at t=1.0 sec.
Simulation to t=20.00 sec completed in 2.4786 seconds.
Outputs to "ieee14_out.lst" and "ieee14_out.npz".
Outputs written in 0.0423 seconds.
```

```
  0%|          | 0/100 [00:00<?, ?%/s]
```

```
True
```

Now let us plot the frequency of the remaining online generators. We multiply by 60 to convert from per-unit to Hz (for a 60 Hz system). Generator GENROU_2 (index 1) was tripped, so we plot only the remaining machines.

```
ss.TDS.load_plotter()

fig, ax = ss.TDS.plt.plot(
    ss.GENROU.omega,
    a=(0, 2, 3, 4),  # Exclude tripped generator (index 1)
    ytimes=60,
    ylabel='Frequency [Hz]'
)
```



The plot shows the classic frequency response to a generation loss: an immediate frequency decline as the generators decelerate, followed by governor action that gradually arrests the decline. The frequency settles at a new steady-state value below nominal (60 Hz) because the governors have droop characteristics rather than isochronous control.

To restore frequency to nominal, either more generation must be added or load must be shed.

2.12.3 Determining the Generation Shortfall

Before implementing load shedding, we need to know how much generation was lost. This information comes from the power flow solution for the tripped generator.

```
# View PV generator data
ss.PV.as_df()[['idx', 'bus', 'p0', 'q0']]
```

```
     idx  bus    p0    q0
uid
0      2    2  0.40  0.15
1      3    3  0.40  0.15
2      4    6  0.30  0.10
3      5    8  0.35  0.10
```

GENROU_2 corresponds to the first PV generator (GENROU_1 corresponds to the Slack bus). The lost active power is 0.40 pu on the 100 MVA system base, which equals 40 MW.

2.12.4 Implementing Load Shedding

We now reload the system and implement a two-stage simulation: first let the frequency decline after the generator trip, then shed 0.4 pu of load at t=2 seconds to compensate for the lost generation.

Load shedding is implemented by reducing the `Ppf` (power flow active power) parameter of selected PQ loads. This is the parameter that the PQ model uses during time-domain simulation when configured for constant power behavior.

```
# Reload the system
ss = andes.load(ieee14_raw, addfile=ieee14_dyr, setup=False)

ss.add("Toggle", model='SynGen', dev="GENROU_2", t=1.0)
ss.setup()
ss.Toggle.u.v[[0, 1]] = 0

ss.PQ.config.p2p = 1
ss.PQ.config.q2q = 1
ss.PQ.config.p2z = 0
ss.PQ.config.q2z = 0
ss.PQ.pq2z = 0

ss.PFlow.run()
```

```
Working directory: "/home/docs/checkouts/readthedocs.org/user_builds/andes/
↪checkouts/stable/docs/source/tutorials"
> Reloaded generated Python code of module "pycode".
Parsing input file "/home/docs/checkouts/readthedocs.org/user_builds/andes/
↪envs/stable/lib/python3.11/site-packages/andes/cases/ieee14/ieee14.raw"...
  IEEE 14 BUS TEST CASE
  03/06/14 CONTO           100.0  1962 W
Input file parsed in 0.0057 seconds.
Parsing additional file "/home/docs/checkouts/readthedocs.org/user_builds/
↪andes/envs/stable/lib/python3.11/site-packages/andes/cases/ieee14/ieee14.dyr
↪"...
Addfile parsed in 0.0824 seconds.
IEEEST <IEEEST_1> added BusFreq <BusFreq_1> linked to bus <3.0>
ST2CUT <ST2CUT_2> added BusFreq <BusFreq_2> linked to bus <1.0>
ST2CUT <ST2CUT_3> added BusFreq <BusFreq_3> linked to bus <2.0>
Connectivity check completed in 0.0001 seconds.
-> System connectivity check results:
  No islanded bus detected.
  System is interconnected.
  Each island has a slack bus correctly defined and enabled.
System internal structure set up in 0.0198 seconds.

-> Power flow calculation
           Numba: Off
   Sparse solver: KLU
 Solution method: NR method
Power flow initialized in 0.0033 seconds.
0: |F(x)| = 0.5605182134
1: |F(x)| = 0.006202200332
2: |F(x)| = 5.819382827e-06
3: |F(x)| = 6.957087684e-12
Converged in 4 iterations in 0.0021 seconds.
Report saved to "ieee14_out.txt" in 0.0006 seconds.
```

```
True
```

```
# Run simulation to t=2 seconds (generator trips at t=1s)
ss.TDS.config.tf = 2.0
ss.TDS.config.criteria = 0
ss.TDS.run()
```

```
DAE compaction: removed 6 algebraic variable slots (m: 246 -> 240)
Initialization for dynamics completed in 0.0258 seconds.
Initialization was successful.
<Toggle Toggle_3>: SynGen.GENROU_2 status changed to 0 at t=1.0 sec.
Simulation to t=2.00 sec completed in 0.3182 seconds.
Outputs to "ieee14_out.lst" and "ieee14_out.npz".
Outputs written in 0.0035 seconds.
```

```
  0%|          | 0/100 [00:00<?, ?%/s]
```

```
True
```

Now we shed load by reducing the active power of loads on selected buses. We distribute the 0.4 pu load reduction equally among six buses.

```
# Select buses for load shedding
shed_buses = [2, 3, 4, 5, 6, 9]

# Find the PQ device indices on these buses
pq_shed_idx = ss.PQ.find_idx(keys='bus', values=shed_buses)
print(f"PQ devices to shed: {pq_shed_idx}")
```

```
PQ devices to shed: ['PQ_1', 'PQ_2', 'PQ_3', 'PQ_4', 'PQ_5', 'PQ_6']
```

```
# Get current active power values
pq_p = ss.PQ.get(src='Ppf', idx=pq_shed_idx, attr='v')
print(f"Current load values: {pq_p}")

# Calculate new values after shedding (distribute 0.4 pu equally)
pq_p_new = pq_p - 0.4 / len(shed_buses)
print(f"New load values: {pq_p_new}")

# Apply the load shedding
ss.PQ.set('Ppf', pq_shed_idx, pq_p_new)
```

```
Current load values: [0.217 0.5   0.478 0.076 0.15  0.295]
New load values: [0.15033333 0.43333333 0.41133333 0.00933333 0.08333333 0.
↪22833333]
```

```
True
```

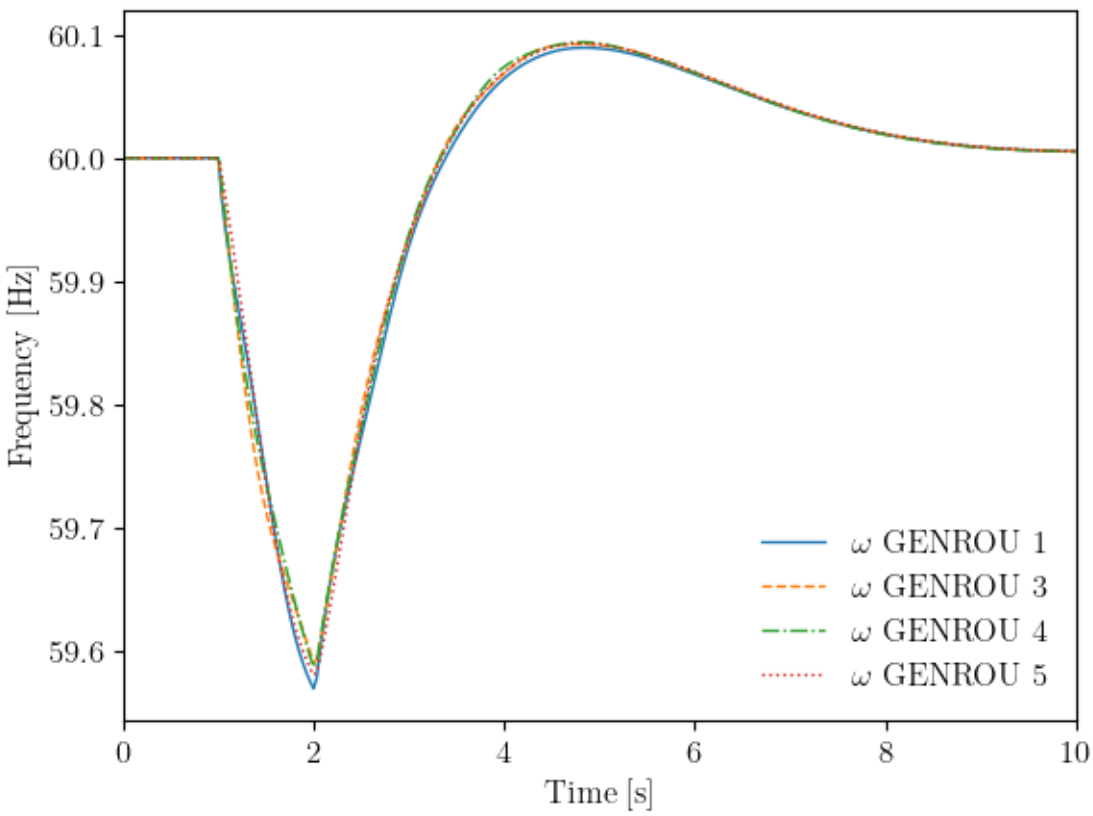
```
# Continue simulation to 10 seconds
ss.TDS.config.tf = 10
ss.TDS.run()
```

```
  0%|          | 0/100 [00:00<?, ?%/s]
```

```
Simulation to t=10.00 sec completed in 1.0669 seconds.
Outputs to "ieee14_out.lst" and "ieee14_out.npz".
Outputs written in 0.0205 seconds.
```

```
True
```

```
fig, ax = ss.TDS.plt.plot(
    ss.GENROU.omega,
    a=(0, 2, 3, 4),
    ytimes=60,
    ylabel='Frequency [Hz]'
)
```



The plot now shows two distinct phases:

1. **t=1-2s**: Frequency declines after the generator trip as the remaining generators decelerate
2. **t=2-10s**: After load shedding at t=2s, frequency recovers back toward 60 Hz

The frequency returns to approximately 60 Hz because we shed exactly the amount of load equal to the lost generation. In practice, under-frequency load shedding (UFLS) schemes are designed with multiple stages that trigger at progressively lower frequencies.

2.12.5 Key Concepts

Concept	Description
Governor droop	Governors respond to frequency deviation proportionally, leading to a new steady-state frequency below nominal
Load shedding	Disconnecting loads to restore generation-load balance
UFLS	Under-Frequency Load Shedding - automated schemes that trip loads at preset frequency thresholds
Rate of Change of Frequency (RoCoF)	How quickly frequency changes after a disturbance, related to system inertia

2.12.6 Cleanup

```
!andes misc -C
```

```
"/home/docs/checkouts/readthedocs.org/user_builds/andes/checkouts/stable/docs/
↪source/tutorials/ieee14_out.npz" removed.
"/home/docs/checkouts/readthedocs.org/user_builds/andes/checkouts/stable/docs/
↪source/tutorials/ieee14_out.txt" removed.
"/home/docs/checkouts/readthedocs.org/user_builds/andes/checkouts/stable/docs/
↪source/tutorials/ieee14_out.lst" removed.
```

2.12.7 Next Steps

- *Eigenvalue Analysis* - Small-signal stability analysis
- *Contingency Analysis* - Systematic contingency screening
- *Dynamic Control and Setpoint Changes* - Runtime parameter modifications

2.13 State Estimation

State estimation (SE) determines the most likely operating state of a power system from a set of noisy, redundant measurements. While power flow computes the exact solution given precise inputs, state estimation works with real-world data that is inherently imprecise. It processes measurements such as bus voltage magnitudes, power injections, and branch flows, and produces the best estimate of all bus voltage magnitudes and angles.

ANDES provides two estimation algorithms. The weighted least squares (WLS) method minimizes the weighted sum of squared measurement residuals and is optimal under Gaussian noise. The least absolute value (LAV) method minimizes the weighted sum of absolute residuals and is robust to gross measurement errors. Redundancy in measurements provides both noise rejection and the ability to detect bad data through statistical tests.

This tutorial covers running state estimation in ANDES, constructing custom measurement sets, interpreting results, performing bad data detection with the chi-squared test, comparing WLS and LAV under bad data conditions, and handling multi-island systems.

```
# Reduce logging verbosity for PDF builds
import os
if os.environ.get('SPHINX_BUILD_PDF'):
    import andes
    _orig_config_logger = andes.config_logger
    def _quiet_logger(stream_level=20, *args, **kwargs):
        stream_level = max(stream_level, 30)
        return _orig_config_logger(stream_level, *args, **kwargs)
    andes.config_logger = _quiet_logger
```

2.13.1 Setup

We begin by importing ANDES, NumPy, and the measurement framework. State estimation requires a converged power flow as the starting point, so we load a case and run power flow first.

```
%matplotlib inline

import numpy as np
import andes
from andes.se import Measurements

andes.config_logger(stream_level=20)
```

```
ss = andes.load(andes.get_case('ieee14/ieee14.json'))
ss.PFlow.run()
```

```
Working directory: "/home/docs/checkouts/readthedocs.org/user_builds/andes/
↪checkouts/stable/docs/source/tutorials"
> Loaded generated Python code in "/home/docs/.andes/pycode".
Parsing input file "/home/docs/checkouts/readthedocs.org/user_builds/andes/
↪envs/stable/lib/python3.11/site-packages/andes/cases/ieee14/ieee14.json"...
Input file parsed in 0.0021 seconds.
Connectivity check completed in 0.0004 seconds.
-> System connectivity check results:
  No islanded bus detected.
  System is interconnected.
  Each island has a slack bus correctly defined and enabled.
System internal structure set up in 0.0211 seconds.

-> Power flow calculation
           Numba: Off
   Sparse solver: KLU
 Solution method: NR method
Power flow initialized in 0.0036 seconds.
0: |F(x)| = 0.5605182134
1: |F(x)| = 0.006202200332
2: |F(x)| = 5.819382827e-06
3: |F(x)| = 6.957087684e-12
Converged in 4 iterations in 0.0022 seconds.
Report saved to "ieee14_out.txt" in 0.0012 seconds.
```

```
True
```

2.13.2 Running State Estimation

The simplest way to run state estimation is to call `ss.SE.run()` with no arguments. ANDES will automatically generate a default set of measurements from the power flow solution, add Gaussian noise, and estimate the system state using the WLS algorithm.

```
ss.SE.run()
```

```
-> State Estimation
          Method: WLS
       Tolerance: 0.0001
  Max iterations: 100
No measurements provided; using default set (all bus voltages + all bus␣
↪injections, 42 measurements).
Adding angle reference at bus 1 for island with 14 buses.
WLS converged in 2 iterations, J = 16.2091
SE converged in 2 iterations in 0.0030 seconds.

-> SE Report
       Converged: True
      Iterations: 2
     Objective J: 16.2091
    Measurements: 43
          States: 28
      Redundancy: 1.54x
        Max |dV|: 0.010248 pu
        Max |da|: 0.0102194 rad
```

```
True
```

The SE report shows convergence status, the number of iterations, the objective function value J, the number of measurements and states, and the measurement redundancy ratio. The estimation error (Max |dV| and Max |da|) compares the estimated state against the power flow solution.

After convergence, the estimated bus voltage magnitudes and angles are available through the `v_est` and `a_est` properties.

```
print(f"Converged: {ss.SE.converged}")
print(f"Estimated voltages (pu): {np.round(ss.SE.v_est, 4)}")
print(f"Estimated angles (rad):  {np.round(ss.SE.a_est, 4)}")
```

```
Converged: True
Estimated voltages (pu): [1.0342 1.0314 1.0067 1.0103 1.016  1.0268 1.0219 1.
↪0264 1.0199 1.0138
 1.0171 1.0103 1.0086 1.0061]
Estimated angles (rad):  [ 0.     -0.0306 -0.0622 -0.0775 -0.0675 -0.1181 -0.
↪0899 -0.0332 -0.1334
 -0.1396 -0.1303 -0.135  -0.1405 -0.1677]
```

2.13.3 Custom Measurements

In practice, not every bus and branch is instrumented. The `Measurements` class allows you to construct a custom measurement set that reflects the actual placement of meters in the system. Each measurement is characterized by its type, location, and standard deviation (sigma), which represents the expected measurement noise.

Four types of measurements are supported:

Method	Measurement type	Description
`add_bus_voltage()`	Voltage magnitude	RTU or PMU voltage measurements
`add_bus_angle()`	Voltage angle	PMU angle measurements
`add_bus_injection()`	P and Q injection	Bus power injection measurements
`add_line_flow()`	P and Q flow	Branch power flow measurements (from-end)

The standard deviation sigma controls the weight of each measurement in the WLS objective. A smaller sigma means the estimator trusts that measurement more. Typical values are 0.005-0.02 for voltage measurements and 0.02-0.05 for power measurements.

```
m = Measurements(ss)

# Voltage meters at all buses (high accuracy)
m.add_bus_voltage(sigma=0.01)

# Power injection meters at all buses
m.add_bus_injection(sigma_p=0.02, sigma_q=0.03)

# Generate noisy measurement values from the PFlow solution
m.generate_from_pflow(seed=42)

print(f"Total measurements: {m.nm}")
print(f"State dimension: {2 * ss.Bus.n}")
print(f"Redundancy: {m.nm / (2 * ss.Bus.n):.1f}x")
```

```
Total measurements: 42
State dimension: 28
Redundancy: 1.5x
```

The redundancy ratio (measurements divided by states) indicates how much extra information is available. A ratio above 1.0 means the system is overdetermined, which is necessary for state estimation to work. Higher redundancy provides better noise rejection and more robust bad data detection. A redundancy of 2-3x is typical for real systems.

Pass the custom measurements to `SE.run()` to use them instead of the default set.

```
ss.SE.run(measurements=m)
```

```
-> State Estimation
          Method: WLS
```

(continues on next page)

(continued from previous page)

```
       Tolerance: 0.0001
  Max iterations: 100
Adding angle reference at bus 1 for island with 14 buses.
WLS converged in 2 iterations, J = 13.027
SE converged in 2 iterations in 0.0021 seconds.

-> SE Report
       Converged: True
      Iterations: 2
     Objective J: 13.027
    Measurements: 43
          States: 28
      Redundancy: 1.54x
        Max |dV|: 0.0059106 pu
        Max |da|: 0.00930009 rad
```

```
True
```

Selective Meter Placement

You can also place measurements on specific buses or lines rather than the entire system. This is useful for modeling realistic SCADA configurations where only key substations have full instrumentation. The `bus_idx` and `line_idx` parameters accept a list of device indices.

```
m2 = Measurements(ss)

# Voltage at selected buses only
m2.add_bus_voltage(bus_idx=[1, 2, 3, 6, 8], sigma=0.005)

# PMU angle measurement at the slack bus
m2.add_bus_angle(bus_idx=[1], sigma=0.01)

# Injection meters at all buses
m2.add_bus_injection(sigma_p=0.02, sigma_q=0.03)

# Flow meters on selected lines
m2.add_line_flow(line_idx=['Line_1', 'Line_2', 'Line_3', 'Line_7', 'Line_8'],
                 sigma_p=0.02, sigma_q=0.03)

m2.generate_from_pflow(seed=42)

print(f"Total measurements: {m2.nm}")
ss.SE.run(measurements=m2)
```

```
Total measurements: 44
```

```
-> State Estimation
          Method: WLS
```

(continues on next page)

(continued from previous page)

```
       Tolerance: 0.0001
  Max iterations: 100
WLS converged in 2 iterations, J = 10.0405
SE converged in 2 iterations in 0.0028 seconds.

-> SE Report
       Converged: True
      Iterations: 2
     Objective J: 10.0405
    Measurements: 44
          States: 28
      Redundancy: 1.57x
        Max |dV|: 0.0082004 pu
        Max |da|: 0.0132314 rad
```

```
True
```

Low-Level `add()` Method

All convenience wrappers delegate to the `add()` method, which accepts a model name and variable name. This method is useful when you need fine-grained control over individual measurements.

```
m3 = Measurements(ss)

# Equivalent to m3.add_bus_voltage(bus_idx=[1, 2], sigma=0.005)
m3.add('Bus', 'v', idx=[1, 2], sigma=0.005)

# Equivalent to m3.add_bus_angle(bus_idx=[1], sigma=0.01)
m3.add('Bus', 'a', idx=[1], sigma=0.01)

print(f"Measurements added: {m3.nm}")
```

```
Measurements added: 3
```

2.13.4 Zero-Noise Verification

A useful sanity check is to verify that state estimation recovers the exact power flow solution when given perfect (noise-free) measurements. In this case, the measurement vector is set to $z = h(x_{\text{true}})$ without any added noise. The WLS estimator should converge to the true state to within numerical precision.

This test validates the correctness of the measurement functions $h(x)$ and the Jacobian $H(x)$.

```
from andes.se import StaticEvaluator

m_exact = Measurements(ss)
m_exact.add_bus_voltage(sigma=0.01)
m_exact.add_bus_angle(sigma=0.01)
```

(continues on next page)

(continued from previous page)

```
m_exact.add_bus_injection(sigma_p=0.02, sigma_q=0.03)
m_exact.add_line_flow(sigma_p=0.02, sigma_q=0.03)

# Set z = h(x_true) exactly (no noise)
m_exact.finalize()
ev = StaticEvaluator(ss, m_exact)
theta_true = np.array(ss.Bus.a.v, dtype=float)
Vm_true = np.array(ss.Bus.v.v, dtype=float)
m_exact.z = ev.h(theta_true, Vm_true)

ss.SE.run(measurements=m_exact)

v_err = np.max(np.abs(ss.SE.v_est - Vm_true))
a_err = np.max(np.abs(ss.SE.a_est - theta_true))
print(f"Max voltage error: {v_err:.2e} pu")
print(f"Max angle error:   {a_err:.2e} rad")
```

```
Max voltage error: 0.00e+00 pu
Max angle error:   0.00e+00 rad
```

```
-> State Estimation
          Method: WLS
       Tolerance: 0.0001
  Max iterations: 100
WLS converged in 1 iterations, J = 0
SE converged in 1 iterations in 0.0022 seconds.

-> SE Report
       Converged: True
      Iterations: 1
     Objective J: 0
    Measurements: 96
          States: 28
      Redundancy: 3.43x
        Max |dV|: 0 pu
        Max |da|: 0 rad
```

The errors are at the level of machine precision ($\sim 10^{-11}$), confirming that the measurement model and solver are implemented correctly.

2.13.5 Bad Data Detection

The chi-squared (χ^2) test is a standard method for detecting the presence of bad data in the measurement set. Under the assumption that measurement errors are Gaussian, the weighted sum of squared residuals $J(\hat{x}) = [z - h(\hat{x})]^T W [z - h(\hat{x})]$ follows a chi-squared distribution with $m - n$ degrees of freedom, where m is the number of measurements and n is the number of states.

If J exceeds the chi-squared threshold at the desired confidence level, the test rejects the null hypothesis that all measurements are normal, indicating possible bad data.

```
# Run SE with normal measurements
m_normal = Measurements(ss)
m_normal.add_bus_voltage(sigma=0.01)
m_normal.add_bus_injection(sigma_p=0.02, sigma_q=0.03)
m_normal.generate_from_pflow(seed=42)

ss.SE.run(measurements=m_normal)

passed, J, threshold, dof = ss.SE.chi_squared_test()
print(f"Chi-squared test: {'PASSED' if passed else 'FAILED'}")
print(f"  J = {J:.4f}, threshold = {threshold:.4f}, dof = {dof}")
```

```
Chi-squared test: PASSED
  J = 13.0270, threshold = 24.9958, dof = 15
```

```
-> State Estimation
          Method: WLS
       Tolerance: 0.0001
  Max iterations: 100
Adding angle reference at bus 1 for island with 14 buses.
WLS converged in 2 iterations, J = 13.027
SE converged in 2 iterations in 0.0021 seconds.

-> SE Report
       Converged: True
      Iterations: 2
     Objective J: 13.027
    Measurements: 43
          States: 28
      Redundancy: 1.54x
        Max |dV|: 0.0059106 pu
        Max |da|: 0.00930009 rad
```

Now we inject a gross error into one measurement (a 10-sigma outlier) and observe the effect on the chi-squared test.

```
# Inject a gross error
m_bad = Measurements(ss)
m_bad.add_bus_voltage(sigma=0.01)
m_bad.add_bus_injection(sigma_p=0.02, sigma_q=0.03)
m_bad.generate_from_pflow(seed=42)

# Corrupt one voltage measurement with a 10-sigma outlier
m_bad.z[0] += 10 * m_bad.sigma[0]

ss.SE.run(measurements=m_bad)

passed, J, threshold, dof = ss.SE.chi_squared_test()
print(f"Chi-squared test: {'PASSED' if passed else 'FAILED'}")
print(f"  J = {J:.4f}, threshold = {threshold:.4f}, dof = {dof}")
```

```
Chi-squared test: FAILED
  J = 109.0202, threshold = 24.9958, dof = 15
```

```
-> State Estimation
          Method: WLS
       Tolerance: 0.0001
  Max iterations: 100
Adding angle reference at bus 1 for island with 14 buses.
WLS converged in 3 iterations, J = 109.02
SE converged in 3 iterations in 0.0033 seconds.

-> SE Report
       Converged: True
      Iterations: 3
     Objective J: 109.02
    Measurements: 43
          States: 28
      Redundancy: 1.54x
        Max |dV|: 0.0148127 pu
        Max |da|: 0.0046609 rad
```

The chi-squared test correctly detects the presence of bad data. In practice, after detection, further analysis (such as the largest normalized residual test) would be used to identify and remove the specific bad measurement.

2.13.6 Robust Estimation with LAV

The weighted least squares estimator is optimal under Gaussian noise but is sensitive to gross measurement errors. The least absolute value (LAV) estimator provides a robust alternative by minimizing the weighted sum of absolute residuals rather than squared residuals:

$$\min_x \sum_i \frac{|z_i - h_i(x)|}{\sigma_i}$$

This L1-norm formulation assigns less influence to measurements with large residuals compared to the L2-norm used by WLS. The LAV problem is solved via iteratively reweighted least squares (IRLS), where each iteration solves a WLS subproblem with weights inversely proportional to the current residual magnitudes. Because IRLS converges more slowly than Gauss-Newton, a higher iteration limit is generally recommended.

The `lav` algorithm function can be passed to `SE.run()` through the `algorithm` parameter. The following example compares WLS and LAV on the corrupted measurement set from the previous section.

```
from andes.se import lav

# True state from the power flow solution
Vm_true = np.array(ss.Bus.v.v, dtype=float)
theta_true = np.array(ss.Bus.a.v, dtype=float)

# WLS errors (from the run in the previous section on m_bad)
wls_v_err = np.max(np.abs(ss.SE.v_est - Vm_true))
wls_a_err = np.max(np.abs(ss.SE.a_est - theta_true))
```

(continues on next page)

(continued from previous page)

```
# Run LAV on the same corrupted measurements
ss.SE.run(measurements=m_bad, algorithm=lav)
lav_v_err = np.max(np.abs(ss.SE.v_est - Vm_true))
lav_a_err = np.max(np.abs(ss.SE.a_est - theta_true))

print(f"{'Metric':<20} {'WLS':>12} {'LAV':>12}")
print(f"{'':-<44}")
print(f"{'Max |dV| (pu)':<20} {wls_v_err:>12.6f} {lav_v_err:>12.6f}")
print(f"{'Max |da| (rad)':<20} {wls_a_err:>12.6f} {lav_a_err:>12.6f}")
```

```
Metric                        WLS          LAV
--------------------------------------------
Max |dV| (pu)            0.014813     0.004423
Max |da| (rad)           0.004661     0.005232
```

```
-> State Estimation
          Method: LAV
       Tolerance: 0.0001
  Max iterations: 100
LAV converged in 34 iterations, J = 27.1812
SE converged in 34 iterations in 0.0223 seconds.

-> SE Report
       Converged: True
      Iterations: 34
     Objective J: 27.1812
    Measurements: 43
          States: 28
      Redundancy: 1.54x
        Max |dV|: 0.00442327 pu
        Max |da|: 0.00523227 rad
```

The LAV estimator produces smaller estimation errors than WLS in the presence of gross measurement errors. The L1-norm formulation effectively downweights the corrupted measurement, preventing it from biasing the state estimate. This robustness comes at the cost of slower convergence, as the IRLS procedure typically requires more iterations than the standard Gauss-Newton method.

Note that the chi-squared test is defined for the WLS objective function and is not applicable after an LAV run. ANDES will issue a warning if `chi_squared_test()` is called following LAV estimation.

2.13.7 Multi-Island Systems

In systems with multiple electrical islands (disconnected components), each island requires its own voltage angle reference because absolute angles are not observable from power measurements alone. ANDES handles this automatically by detecting islands through connectivity analysis and adding a pseudo-measurement for the angle reference at the slack bus of each island.

The Kundur two-area system provides a convenient example when configured as two separate islands.

```
ss_island = andes.load(andes.get_case('kundur/kundur_islands.xlsx'),
                       default_config=True)
ss_island.PFlow.run()

print(f"Number of islands: {len(ss_island.Bus.island_sets)}")
```

```
Number of islands: 2
```

```
Working directory: "/home/docs/checkouts/readthedocs.org/user_builds/andes/
↪checkouts/stable/docs/source/tutorials"
> Reloaded generated Python code of module "pycode".
Parsing input file "/home/docs/checkouts/readthedocs.org/user_builds/andes/
↪envs/stable/lib/python3.11/site-packages/andes/cases/kundur/kundur_islands.
↪xlsx"...
Input file parsed in 0.0322 seconds.
Connectivity check completed in 0.0001 seconds.
-> System connectivity check results:
  No islanded bus detected.
  System contains 2 island(s).
  Each island has a slack bus correctly defined and enabled.
System internal structure set up in 0.0199 seconds.

-> Power flow calculation
           Numba: Off
   Sparse solver: KLU
 Solution method: NR method
Power flow initialized in 0.0031 seconds.
0: |F(x)| = 14.9282832
1: |F(x)| = 3.670045871
2: |F(x)| = 0.1542941496
3: |F(x)| = 0.0003701017904
4: |F(x)| = 2.966633161e-09
Converged in 5 iterations in 0.0025 seconds.
Report saved to "kundur_islands_out.txt" in 0.0007 seconds.
```

```
m_island = Measurements(ss_island)
m_island.add_bus_voltage(sigma=0.01)
m_island.add_bus_injection(sigma_p=0.02, sigma_q=0.03)
m_island.generate_from_pflow(seed=42)

ss_island.SE.run(measurements=m_island)
```

```
-> State Estimation
          Method: WLS
       Tolerance: 0.0001
  Max iterations: 100
Adding angle reference at bus 1 for island with 5 buses.
Adding angle reference at bus 3 for island with 5 buses.
WLS converged in 2 iterations, J = 8.07042
SE converged in 2 iterations in 0.0022 seconds.

-> SE Report
       Converged: True
      Iterations: 2
     Objective J: 8.07042
    Measurements: 32
          States: 20
      Redundancy: 1.60x
        Max |dV|: 0.00799312 pu
        Max |da|: 0.00501073 rad
```

```
True
```

The log shows that ANDES added angle references for both islands automatically. Without these references, the WLS gain matrix would be singular and the estimator would fail.

2.13.8 Configuration

The SE routine provides several configuration options accessible through `ss.SE.config`.

```
ss.SE.config
```

```
OrderedDict([('linsolve', 0),
             ('tol', 0.0001),
             ('max_iter', 100),
             ('flat_start', 0),
             ('report', 1)])
```

Option	Default	Description
`tol`	1e-4	Convergence tolerance for the SE iteration
`max_iter`	100	Maximum number of iterations
`flat_start`	0	Use flat start (1.0 pu, 0 rad) instead of PFlow solution
`report`	1	Print result summary after estimation

The flat start option is useful for testing robustness of convergence when the power flow solution is not available as an initial guess.

```
# Run with flat start
ss.SE.config.flat_start = 1
```

(continues on next page)

(continued from previous page)

```
ss.SE.run(measurements=m_normal)

# Reset to default
ss.SE.config.flat_start = 0
```

```
-> State Estimation
          Method: WLS
       Tolerance: 0.0001
  Max iterations: 100
WLS converged in 3 iterations, J = 13.027
SE converged in 3 iterations in 0.0026 seconds.

-> SE Report
       Converged: True
      Iterations: 3
     Objective J: 13.027
    Measurements: 43
          States: 28
      Redundancy: 1.54x
        Max |dV|: 0.00591063 pu
        Max |da|: 0.00930006 rad
```

2.13.9 Cleanup

```
!andes misc -C
```

```
"/home/docs/checkouts/readthedocs.org/user_builds/andes/checkouts/stable/docs/
↪source/tutorials/kundur_islands_out.txt" removed.
"/home/docs/checkouts/readthedocs.org/user_builds/andes/checkouts/stable/docs/
↪source/tutorials/ieee14_out.txt" removed.
```

2.13.10 Next Steps

- *Eigenvalue Analysis* - Small-signal stability assessment
- *Contingency Analysis* - N-1 contingency screening

2.14 Continuation Power Flow

Standard power flow computes the operating point at a given loading level, but it cannot determine how far the system is from voltage collapse. Continuation power flow (CPF) answers this question by tracing the relationship between loading and bus voltages, known as the *nose curve* or *PV curve*, from the base case up to and beyond the maximum loadability point.

The CPF uses a predictor-corrector method with a continuation parameter λ that interpolates between the base case ($\lambda = 0$) and a target loading condition ($\lambda = 1$):

$$p(\lambda) = p_{\text{base}} + \lambda \cdot (p_{\text{target}} - p_{\text{base}})$$

At each step, a tangent predictor estimates the next point on the curve, and an augmented Newton-Raphson corrector refines it. The maximum λ achieved gives the steady-state loadability limit, the point beyond which no power flow solution exists.

This tutorial covers running CPF in ANDES, interpreting results, plotting PV curves, QV curve analysis for reactive power margins, selecting parameterization methods, and specifying custom loading targets.

```
%matplotlib inline

# Reduce logging verbosity for PDF builds
import os
if os.environ.get('SPHINX_BUILD_PDF'):
    import andes
    _orig_config_logger = andes.config_logger
    def _quiet_logger(stream_level=20, *args, **kwargs):
        stream_level = max(stream_level, 30)
        return _orig_config_logger(stream_level, *args, **kwargs)
    andes.config_logger = _quiet_logger
```

2.14.1 Setup

CPF requires a converged power flow as its starting point. We load the IEEE 14-bus system and run power flow first.

```
import numpy as np
import andes
from matplotlib import pyplot as plt

andes.config_logger(stream_level=20)
```

```
ss = andes.load(andes.get_case('ieee14/ieee14.json'))
ss.PFlow.run()
```

```
Working directory: "/home/docs/checkouts/readthedocs.org/user_builds/andes/
↪checkouts/stable/docs/source/tutorials"
> Loaded generated Python code in "/home/docs/.andes/pycode".
Parsing input file "/home/docs/checkouts/readthedocs.org/user_builds/andes/
↪envs/stable/lib/python3.11/site-packages/andes/cases/ieee14/ieee14.json"...
Input file parsed in 0.0020 seconds.
Connectivity check completed in 0.0001 seconds.
-> System connectivity check results:
  No islanded bus detected.
  System is interconnected.
  Each island has a slack bus correctly defined and enabled.
System internal structure set up in 0.0198 seconds.
```

(continues on next page)

(continued from previous page)

```
-> Power flow calculation
           Numba: Off
   Sparse solver: KLU
 Solution method: NR method
Power flow initialized in 0.0033 seconds.
0: |F(x)| = 0.5605182134
1: |F(x)| = 0.006202200332
2: |F(x)| = 5.819382827e-06
3: |F(x)| = 6.957087684e-12
Converged in 4 iterations in 0.0022 seconds.
Report saved to "ieee14_out.txt" in 0.0010 seconds.
```

```
True
```

2.14.2 Running CPF

The simplest way to run CPF is to specify a uniform load scaling factor. A `load_scale` of 2.0 means the target loading is twice the base case: all PQ loads and PV generation are scaled proportionally. The CPF traces the nose curve from $\lambda = 0$ (base case) toward $\lambda = 1$ (target), and beyond if the system can sustain it.

```
ss.CPF.run(load_scale=2.0)
```

```
-> Continuation Power Flow
Parameterization: pseudo_arclength
       Step size: 0.1
   Adaptive step: On
         Stop at: NOSE
Nose detected at step 16, lambda=3.167739

-> CPF Report
       Converged: True
           Steps: 17
      Max lambda: 3.258046
     Termination: Nose point at lambda=3.258046
CPF completed in 17 steps in 0.0374 seconds. max lambda = 3.258046
```

```
True
```

The CPF report shows the number of continuation steps, the maximum λ reached (the loadability limit), and the termination reason. A `max_lam` greater than 1.0 means the system can sustain the full target loading; the nose point occurs at some higher loading level.

The `run()` method returns `True` if the CPF terminated as expected for the requested `stop_at` mode:

- `stop_at='NOSE'` (default): terminates at the nose point (maximum loadability)
- `stop_at='FULL'`: traces both the upper and lower branches, returning to $\lambda = 0$
- `stop_at=float`: terminates when λ reaches the specified value

The key results are stored as NumPy arrays:

Attribute	Shape	Description
`ss.CPF.lam`	`(npoints,)`	Continuation parameter λ at each step
`ss.CPF.V`	`(nbus, npoints)`	Bus voltage magnitudes at each step
`ss.CPF.theta`	`(nbus, npoints)`	Bus voltage angles at each step
`ss.CPF.max_lam`	scalar	Maximum λ (loadability limit)
`ss.CPF.steps`	`(npoints-1,)`	Step sizes used at each continuation step

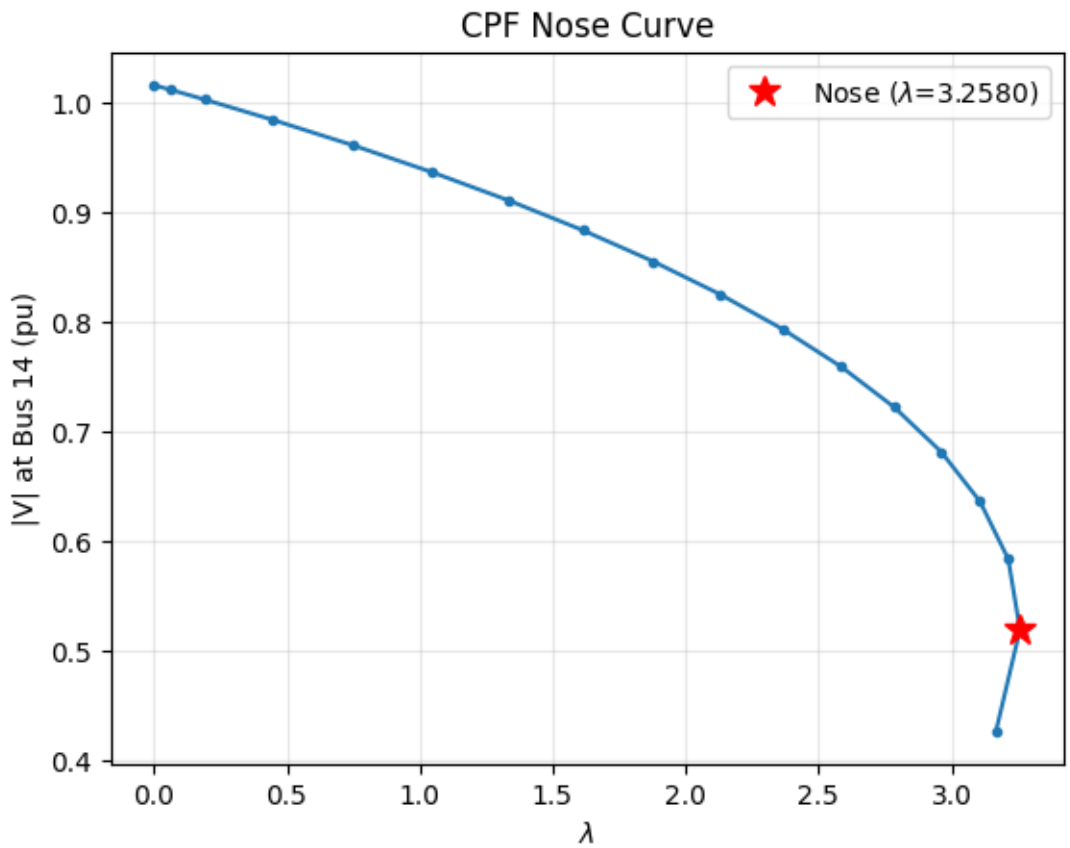
```
ss.CPF.max_lam
```

```
3.2580458612253804
```

2.14.3 Plotting the Nose Curve

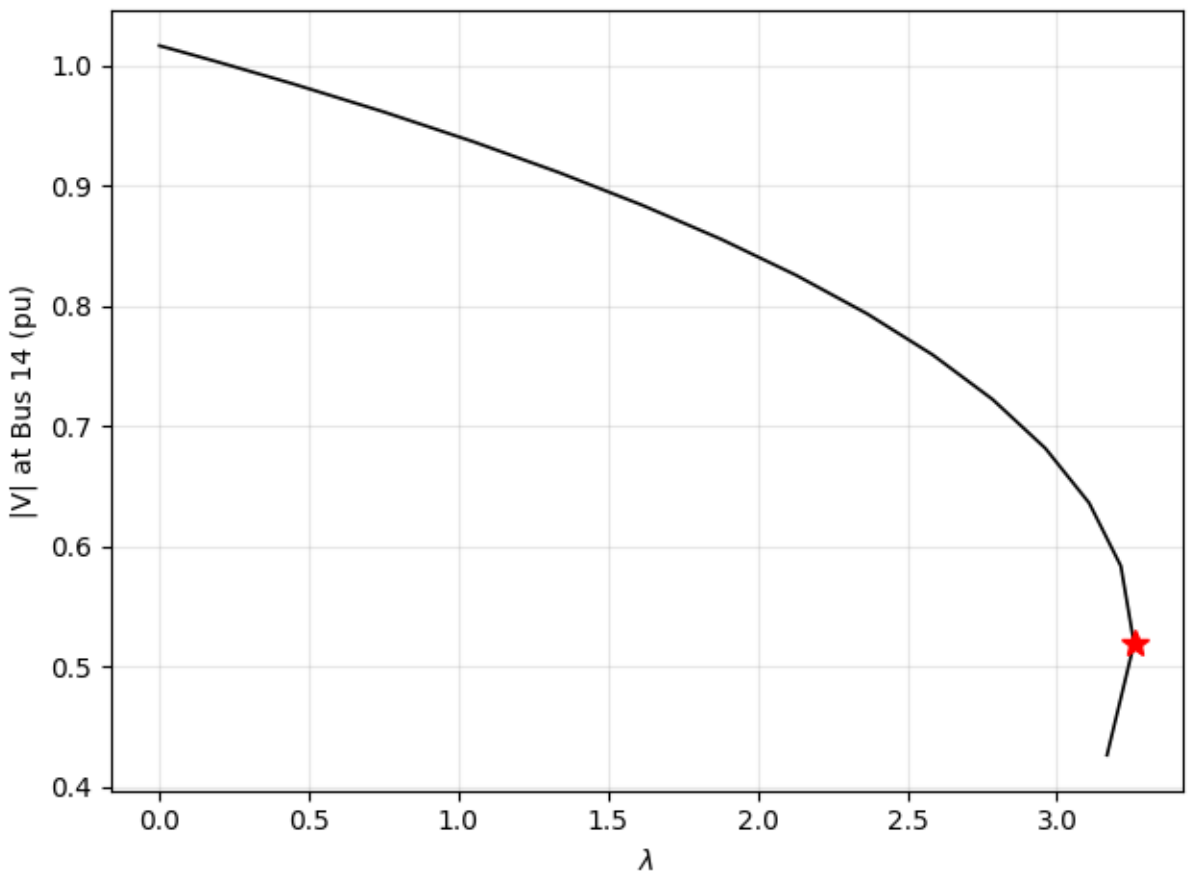
The PV curve reveals how bus voltages change with loading. Weak buses, those with the steepest voltage decline, are the most vulnerable to voltage collapse. The built-in `plot()` method creates a PV curve for a specified bus, with the nose point marked.

```
fig, ax = ss.CPF.plot(bus_idx=14, show=False)
plt.show()
```

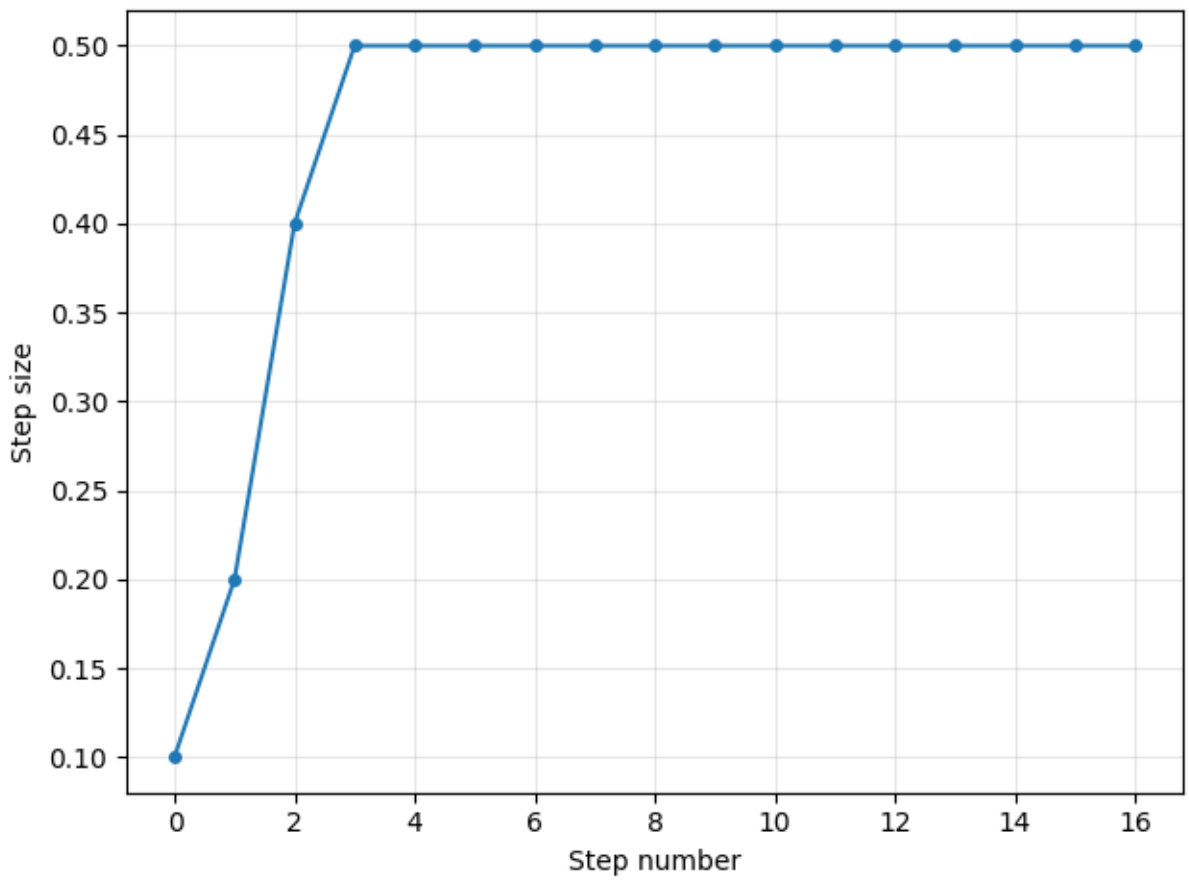


The `V`, `lam`, and `theta` arrays can be used directly for custom analysis. The following example extracts the nose-point data, identifies the weakest bus, and plots a publication-quality PV curve.

```
lam = ss.CPF.lam
nose = np.argmax(lam)

# Weakest bus = largest voltage drop from base to nose.
weak = np.argmax(ss.CPF.V[:, 0] - ss.CPF.V[:, nose])

fig, ax = plt.subplots()
ax.plot(lam, ss.CPF.V[weak, :], 'k-', linewidth=1.2)
ax.plot(lam[nose], ss.CPF.V[weak, nose], 'r*', markersize=10)
ax.set_xlabel(r'$\lambda$')
ax.set_ylabel(f'|V| at Bus {ss.Bus.idx.v[weak]} (pu)')
ax.grid(True, alpha=0.3)
fig.tight_layout()
plt.show()
```




```
fig, ax = plt.subplots()
ax.plot(range(len(ss.CPF.steps)), ss.CPF.steps, 'o-', markersize=4)
ax.set_xlabel('Step number')
ax.set_ylabel('Step size')
ax.grid(True, alpha=0.3)
fig.tight_layout()
plt.show()
```



2.14.4 Parameterization Methods

The parameterization determines how the corrector constrains the solution at each step. Different methods have different trade-offs near the nose point, where the standard power flow Jacobian becomes singular.

ANDES provides three parameterization methods:

Method	Description	Behavior at nose
`pseudo_arclenc`	Constrains the step along the tangent direction	Most robust; can trace through the nose point
`arclength`	Constrains the Euclidean distance from the previous point	Traces the full curve; may take more steps
`natural`	Fixes λ increment directly	Simplest; fails at the nose point

Pseudo arc-length is the default because it handles the nose point most reliably. Natural parameterization is the simplest conceptually but becomes singular exactly at the loadability limit, making it unable to trace the lower branch of the PV curve.

The parameterization is selected through the configuration before running CPF.

```
# reload to get a fresh system
ss2 = andes.load(andes.get_case('ieee14/ieee14.json'))
ss2.PFlow.run()

ss2.CPF.config.parameterization = 'arclength'
ss2.CPF.run(load_scale=2.0)
```

```
Working directory: "/home/docs/checkouts/readthedocs.org/user_builds/andes/
↪checkouts/stable/docs/source/tutorials"
> Reloaded generated Python code of module "pycode".
Parsing input file "/home/docs/checkouts/readthedocs.org/user_builds/andes/
↪envs/stable/lib/python3.11/site-packages/andes/cases/ieee14/ieee14.json"...
Input file parsed in 0.0019 seconds.
Connectivity check completed in 0.0001 seconds.
-> System connectivity check results:
  No islanded bus detected.
  System is interconnected.
  Each island has a slack bus correctly defined and enabled.
System internal structure set up in 0.0190 seconds.

-> Power flow calculation
           Numba: Off
   Sparse solver: KLU
 Solution method: NR method
Power flow initialized in 0.0032 seconds.
0: |F(x)| = 0.5605182134
1: |F(x)| = 0.006202200332
2: |F(x)| = 5.819382827e-06
3: |F(x)| = 6.957087684e-12
Converged in 4 iterations in 0.0020 seconds.
Report saved to "ieee14_out.txt" in 0.0006 seconds.

-> Continuation Power Flow
Parameterization: arclength
       Step size: 0.1
   Adaptive step: On
         Stop at: NOSE
Nose detected at step 16, lambda=3.175750

-> CPF Report
       Converged: True
           Steps: 17
      Max lambda: 3.258050
     Termination: Nose point at lambda=3.258050
CPF completed in 17 steps in 0.0384 seconds. max lambda = 3.258050
```

```
True
```

```
ss2.CPF.max_lam
```

```
3.2580499020258302
```

All three methods should agree on the loadability limit to within a few percent, since they trace the same underlying curve. The differences arise from how finely they sample the curve near the nose.

2.14.5 Termination Control

By default, CPF terminates at the nose point (`stop_at='NOSE'`). In some analyses, it is useful to stop at a specific loading level instead, for example, to find the voltage profile at 1.5 times the base-case loading. Setting `stop_at` to a float value causes CPF to terminate when λ reaches that value, with the final step sized to land exactly on the target.

```
ss3 = andes.load(andes.get_case('ieee14/ieee14.json'))
ss3.PFlow.run()

ss3.CPF.config.stop_at = 1.5
ss3.CPF.run(load_scale=2.0)
```

```
Working directory: "/home/docs/checkouts/readthedocs.org/user_builds/andes/
↪checkouts/stable/docs/source/tutorials"
> Reloaded generated Python code of module "pycode".
Parsing input file "/home/docs/checkouts/readthedocs.org/user_builds/andes/
↪envs/stable/lib/python3.11/site-packages/andes/cases/ieee14/ieee14.json"...
Input file parsed in 0.0019 seconds.
Connectivity check completed in 0.0001 seconds.
-> System connectivity check results:
  No islanded bus detected.
  System is interconnected.
  Each island has a slack bus correctly defined and enabled.
System internal structure set up in 0.0186 seconds.

-> Power flow calculation
           Numba: Off
   Sparse solver: KLU
 Solution method: NR method
Power flow initialized in 0.0032 seconds.
0: |F(x)| = 0.5605182134
1: |F(x)| = 0.006202200332
2: |F(x)| = 5.819382827e-06
3: |F(x)| = 6.957087684e-12
Converged in 4 iterations in 0.0019 seconds.
Report saved to "ieee14_out.txt" in 0.0006 seconds.

-> Continuation Power Flow
Parameterization: pseudo_arclength
       Step size: 0.1
   Adaptive step: On
         Stop at: 1.5

-> CPF Report
```

(continues on next page)

(continued from previous page)

```
       Converged: True
           Steps: 8
      Max lambda: 1.500000
     Termination: Reached target lambda=1.5
CPF completed in 8 steps in 0.0152 seconds. max lambda = 1.500000
```

```
True
```

```
ss3.CPF.lam[-1]
```

```
np.float64(1.5)
```

The final λ value is exactly the requested target (to within solver tolerance).

2.14.6 Full Curve Tracing (Upper and Lower Branches)

Setting `stop_at='FULL'` traces the complete nose curve: the upper branch from $\lambda = 0$ to the nose, then the lower (unstable) branch back down to $\lambda = 0$. At the nose point, the continuation detects the fold bifurcation and automatically switches to the lower branch. The lower branch represents unstable equilibrium points that are physically unrealizable but are important for voltage stability analysis.

The lower branch voltages are significantly below the upper branch at the same loading level. At $\lambda = 0$ on the lower branch, the bus voltages represent the low-voltage solution of the base-case power flow equations.

```
ss_full = andes.load(andes.get_case('ieee14/ieee14.json'))
ss_full.PFlow.run()

ss_full.CPF.config.stop_at = 'FULL'
ss_full.CPF.run(load_scale=2.0)
```

```
Working directory: "/home/docs/checkouts/readthedocs.org/user_builds/andes/
↪checkouts/stable/docs/source/tutorials"
> Reloaded generated Python code of module "pycode".
Parsing input file "/home/docs/checkouts/readthedocs.org/user_builds/andes/
↪envs/stable/lib/python3.11/site-packages/andes/cases/ieee14/ieee14.json"...
Input file parsed in 0.0021 seconds.
Connectivity check completed in 0.0001 seconds.
-> System connectivity check results:
  No islanded bus detected.
  System is interconnected.
  Each island has a slack bus correctly defined and enabled.
System internal structure set up in 0.0206 seconds.

-> Power flow calculation
           Numba: Off
   Sparse solver: KLU
 Solution method: NR method
Power flow initialized in 0.0035 seconds.
```

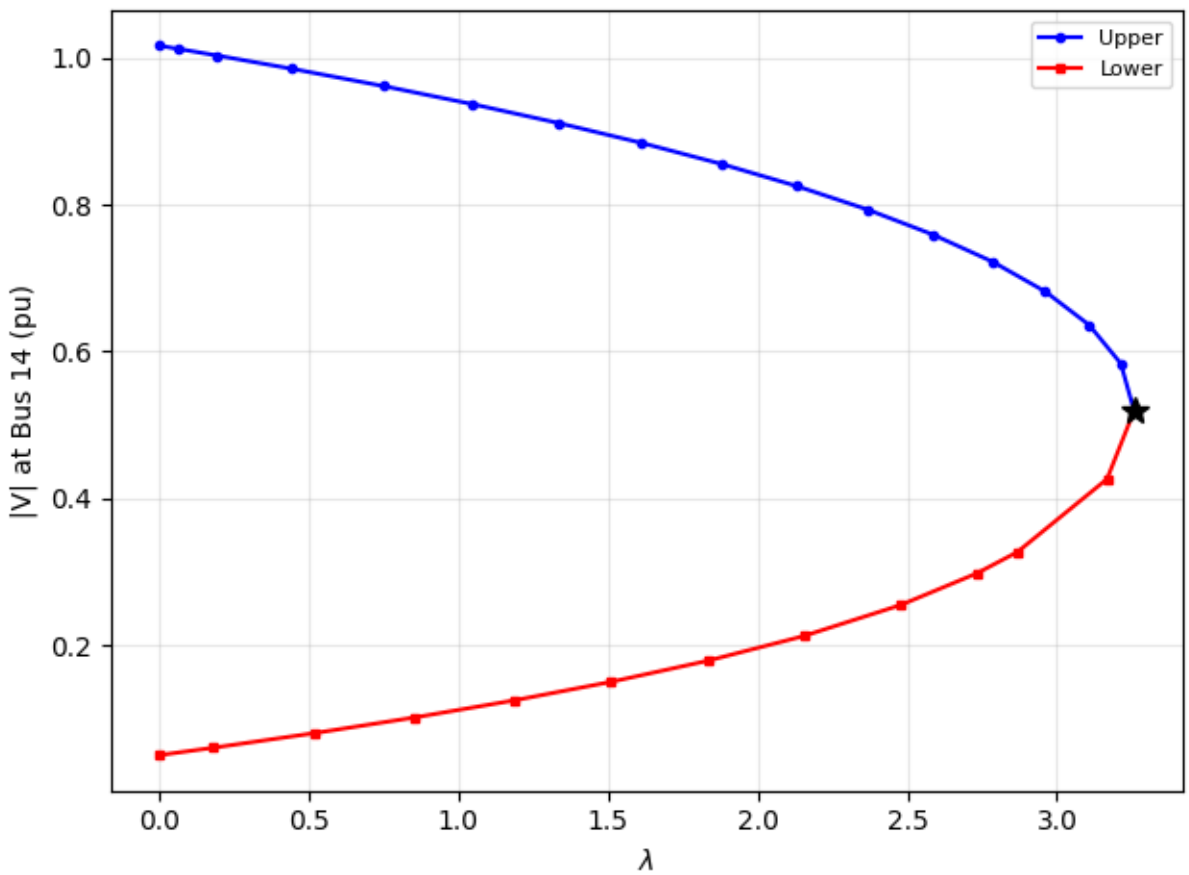
(continues on next page)

(continued from previous page)

```
0: |F(x)| = 0.5605182134
1: |F(x)| = 0.006202200332
2: |F(x)| = 5.819382827e-06
3: |F(x)| = 6.957087684e-12
Converged in 4 iterations in 0.0020 seconds.
Report saved to "ieee14_out.txt" in 0.0006 seconds.

-> Continuation Power Flow
Parameterization: pseudo_arclength
       Step size: 0.1
   Adaptive step: On
         Stop at: FULL
Nose detected at step 16, lambda=3.167739

-> CPF Report
       Converged: True
           Steps: 28
      Max lambda: 3.258046
     Termination: Full curve traced (returned to lambda=0)
CPF completed in 28 steps in 0.0620 seconds. max lambda = 3.258046
```

```
True
```

```
lam = ss_full.CPF.lam
nose = np.argmax(lam)
weak = np.argmax(ss_full.CPF.V[:, 0] - ss_full.CPF.V[:, nose])

fig, ax = plt.subplots()
ax.plot(lam[:nose+1], ss_full.CPF.V[weak, :nose+1], 'b-o', markersize=3,␣
↪label='Upper')
ax.plot(lam[nose:], ss_full.CPF.V[weak, nose:], 'r-s', markersize=3, label=
↪'Lower')
ax.plot(lam[nose], ss_full.CPF.V[weak, nose], 'k*', markersize=10)
ax.set_xlabel(r'$\lambda$')
ax.set_ylabel(f'|V| at Bus {ss_full.Bus.idx.v[weak]} (pu)')
ax.legend(fontsize=8)
ax.grid(True, alpha=0.3)
fig.tight_layout()
plt.show()
```



2.14.7 Per-Device Loading Targets

Uniform load scaling is the most common use case, but real planning studies often involve non-uniform load growth. For example, one area may grow faster than another. The `p0_target` and `q0_target` arguments accept per-device arrays that specify the target active and reactive power for each PQ load. Similarly, `pg_target` specifies the target generation for each PV generator.

The continuation parameter λ then interpolates between the base-case values and these per-device targets.

```
ss4 = andes.load(andes.get_case('ieee14/ieee14.json'))
ss4.PFlow.run()

# Double the load on the first 5 PQ buses, keep the rest unchanged
p0_base = ss4.PQ.p0.v.copy()
q0_base = ss4.PQ.q0.v.copy()

p0_target = p0_base.copy()
q0_target = q0_base.copy()
p0_target[:5] = p0_base[:5] * 3.0
q0_target[:5] = q0_base[:5] * 3.0

ss4.CPF.run(p0_target=p0_target, q0_target=q0_target)
```

```
Working directory: "/home/docs/checkouts/readthedocs.org/user_builds/andes/
↪checkouts/stable/docs/source/tutorials"
> Reloaded generated Python code of module "pycode".
Parsing input file "/home/docs/checkouts/readthedocs.org/user_builds/andes/
↪envs/stable/lib/python3.11/site-packages/andes/cases/ieee14/ieee14.json"...
Input file parsed in 0.0020 seconds.
Connectivity check completed in 0.0001 seconds.
-> System connectivity check results:
  No islanded bus detected.
  System is interconnected.
  Each island has a slack bus correctly defined and enabled.
System internal structure set up in 0.0187 seconds.

-> Power flow calculation
           Numba: Off
   Sparse solver: KLU
 Solution method: NR method
Power flow initialized in 0.0032 seconds.
0: |F(x)| = 0.5605182134
1: |F(x)| = 0.006202200332
2: |F(x)| = 5.819382827e-06
3: |F(x)| = 6.957087684e-12
Converged in 4 iterations in 0.0020 seconds.
Report saved to "ieee14_out.txt" in 0.0006 seconds.

-> Continuation Power Flow
Parameterization: pseudo_arclength
       Step size: 0.1
   Adaptive step: On
         Stop at: NOSE
Nose detected at step 56, lambda=3.516561

-> CPF Report
       Converged: True
           Steps: 57
      Max lambda: 3.517463
     Termination: Nose point at lambda=3.517463
CPF completed in 57 steps in 0.0992 seconds. max lambda = 3.517463
```

```
True
```

```
ss4.CPF.max_lam
```

```
3.5174627656369064
```

2.14.8 QV Curve Analysis

A QV curve shows the reactive power required to maintain a given voltage at a specific bus. The nose of the QV curve gives the reactive power margin, the additional reactive power absorption the bus can sustain before voltage collapse. While the PV curve parameterizes active power, the QV curve fixes active power everywhere and varies only reactive power at the target bus.

The `run_qv()` method builds the appropriate continuation target internally and delegates to the CPF engine. The `q_range` argument specifies the total reactive power sweep range in per-unit on the system base.

```
ss_qv = andes.load(andes.get_case('ieee14/ieee14.json'))
ss_qv.PFlow.run()

ss_qv.CPF.run_qv(bus_idx=14, q_range=3.0)
```

```
Working directory: "/home/docs/checkouts/readthedocs.org/user_builds/andes/
↪checkouts/stable/docs/source/tutorials"
> Reloaded generated Python code of module "pycode".
Parsing input file "/home/docs/checkouts/readthedocs.org/user_builds/andes/
↪envs/stable/lib/python3.11/site-packages/andes/cases/ieee14/ieee14.json"...
Input file parsed in 0.0020 seconds.
Connectivity check completed in 0.0001 seconds.
-> System connectivity check results:
  No islanded bus detected.
  System is interconnected.
  Each island has a slack bus correctly defined and enabled.
System internal structure set up in 0.0187 seconds.

-> Power flow calculation
           Numba: Off
   Sparse solver: KLU
 Solution method: NR method
Power flow initialized in 0.0032 seconds.
0: |F(x)| = 0.5605182134
1: |F(x)| = 0.006202200332
2: |F(x)| = 5.819382827e-06
3: |F(x)| = 6.957087684e-12
Converged in 4 iterations in 0.0021 seconds.
Report saved to "ieee14_out.txt" in 0.0006 seconds.

-> Continuation Power Flow
Parameterization: pseudo_arclength
       Step size: 0.1
   Adaptive step: On
         Stop at: NOSE
Nose detected at step 5, lambda=0.346459

-> CPF Report
       Converged: True
           Steps: 6
      Max lambda: 0.359814
     Termination: Nose point at lambda=0.359814
```

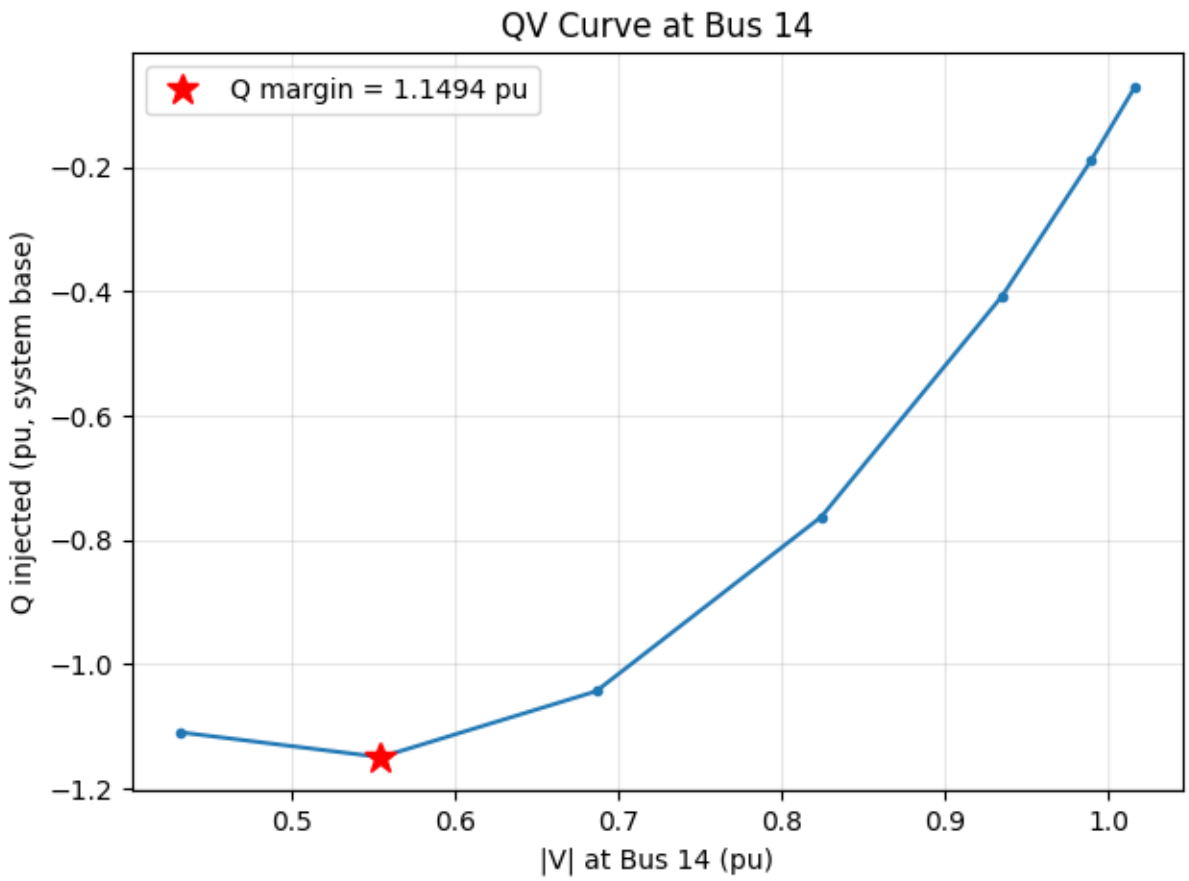
(continues on next page)

(continued from previous page)

```
CPF completed in 6 steps in 0.0131 seconds. max lambda = 0.359814
```

```
True
```

```
fig, ax = ss_qv.CPF.plot_qv(show=False)
fig.tight_layout()
plt.show()
```



The QV curve results are stored in `qv_q` (reactive power in load convention) and `qv_v` (voltage magnitude), both 1-D arrays aligned with `lam`. The `plot_qv()` method negates `qv_q` to follow the standard convention where positive Q represents injection (capacitive support). The red star marks the nose point, which indicates the maximum reactive power that can be absorbed at the bus before voltage collapse.

2.14.9 Configuration

The CPF routine provides several configuration options accessible through `ss.CPF.config`.

```
ss.CPF.config
```

```
OrderedDict([('linsolve', 0),
             ('parameterization', 'pseudo_arclength'),
             ('step', 0.1),
             ('step_min', 0.0001),
             ('step_max', 0.5),
             ('adapt_step', 1),
             ('tol', 1e-06),
             ('max_iter', 20),
             ('max_steps', 500),
             ('stop_at', 'NOSE'),
             ('report', 1)])
```

Option	Default	Description
`parameterization`	`pseudo_arclenc`	Continuation method: `pseudo_arclength`, `arclength`, or `natural`
`step`	0.1	Initial step size for the continuation parameter
`step_min`	1e-4	Minimum step size before declaring convergence failure
`step_max`	0.5	Maximum step size (limits adaptive growth)
`adapt_step`	1	Enable adaptive step sizing (grow on easy steps, shrink on hard)
`tol`	1e-6	Convergence tolerance for the augmented NR corrector
`max_iter`	20	Maximum Newton-Raphson iterations per corrector step
`max_steps`	500	Maximum number of continuation steps
`stop_at`	`NOSE`	Termination: `'NOSE'`, `'FULL'` (trace lower branch), or a float λ
`report`	1	Print summary report after completion

Reactive power limits are enforced through the PV model's built-in PV-to-PQ conversion. Pass `config_option=["PV.pv2pq=1"]` to `andes.load()` to enable Q-limit checking at each corrector step. See the section below for a demonstration.

2.14.10 Reactive Power Limits

When `PV.config.pv2pq` is enabled, generators that hit their reactive power limits are converted from voltage control (PV) to fixed reactive power (PQ) at each corrector step. This reduces the maximum loadability because fewer generators maintain voltage support.

The following example compares the nose curve with and without Q-limit enforcement on the IEEE 14-bus system. The `pv2pq` setting must be passed at load time via `config_option` so that the internal limiter is initialized as enabled.

```
# Without Q limits (default)
ss_noq = andes.load(andes.get_case('ieee14/ieee14.json'))
ss_noq.PFlow.run()
ss_noq.CPF.run(load_scale=2.0)

# With Q limits — must be set at load time so the limiter is initialized␣
↪enabled
ss_qlim = andes.load(andes.get_case('ieee14/ieee14.json'),
                     config_option=["PV.pv2pq=1"])
ss_qlim.PFlow.run()
ss_qlim.CPF.run(load_scale=2.0)

print(f"Max lambda without Q limits: {ss_noq.CPF.max_lam:.4f}")
print(f"Max lambda with Q limits:    {ss_qlim.CPF.max_lam:.4f}")
```

```
Max lambda without Q limits: 3.2580
Max lambda with Q limits:    0.7536
```

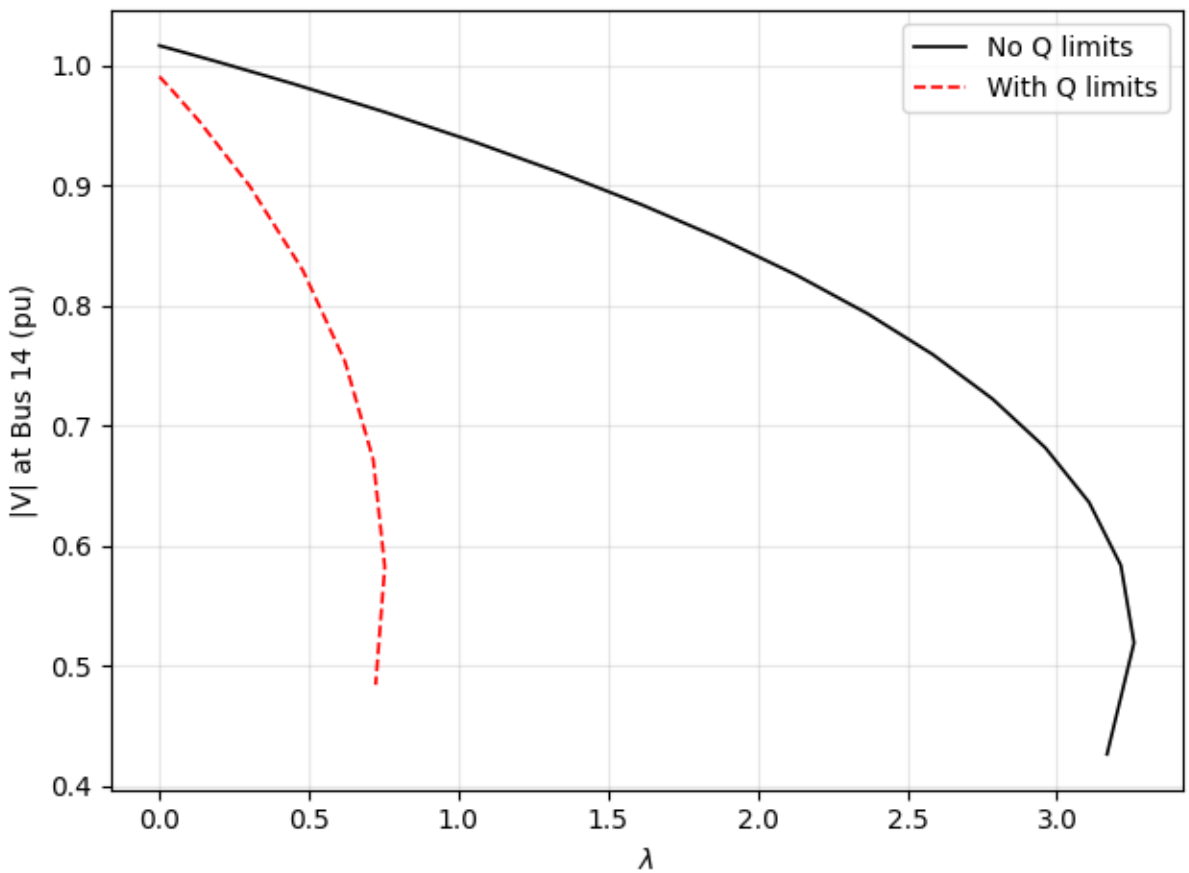
```
Working directory: "/home/docs/checkouts/readthedocs.org/user_builds/andes/
↪checkouts/stable/docs/source/tutorials"
> Reloaded generated Python code of module "pycode".
Parsing input file "/home/docs/checkouts/readthedocs.org/user_builds/andes/
↪envs/stable/lib/python3.11/site-packages/andes/cases/ieee14/ieee14.json"...
Input file parsed in 0.0020 seconds.
Connectivity check completed in 0.0001 seconds.
-> System connectivity check results:
  No islanded bus detected.
  System is interconnected.
  Each island has a slack bus correctly defined and enabled.
System internal structure set up in 0.0191 seconds.

-> Power flow calculation
           Numba: Off
   Sparse solver: KLU
 Solution method: NR method
Power flow initialized in 0.0032 seconds.
0: |F(x)| = 0.5605182134
1: |F(x)| = 0.006202200332
2: |F(x)| = 5.819382827e-06
3: |F(x)| = 6.957087684e-12
Converged in 4 iterations in 0.0020 seconds.
Report saved to "ieee14_out.txt" in 0.0006 seconds.

-> Continuation Power Flow
Parameterization: pseudo_arclength
       Step size: 0.1
   Adaptive step: On
         Stop at: NOSE
Nose detected at step 16, lambda=3.167739

-> CPF Report
       Converged: True
```

(continues on next page)

(continued from previous page)

```
            Steps: 17
       Max lambda: 3.258046
      Termination: Nose point at lambda=3.258046
CPF completed in 17 steps in 0.0361 seconds. max lambda = 3.258046
Working directory: "/home/docs/checkouts/readthedocs.org/user_builds/andes/
↪checkouts/stable/docs/source/tutorials"
> Reloaded generated Python code of module "pycode".
Generated code for <PV> is stale.
Numerical code generation (rapid incremental mode) started...
Generating code for 1 models on 2 processes.
Saved generated pycode to "/home/docs/.andes/pycode"
> Reloaded generated Python code of module "pycode".
Generated numerical code for 1 models in 0.1376 seconds.
Parsing input file "/home/docs/checkouts/readthedocs.org/user_builds/andes/
↪envs/stable/lib/python3.11/site-packages/andes/cases/ieee14/ieee14.json"...
Input file parsed in 0.0021 seconds.
Connectivity check completed in 0.0001 seconds.
-> System connectivity check results:
  No islanded bus detected.
  System is interconnected.
  Each island has a slack bus correctly defined and enabled.
System internal structure set up in 0.0219 seconds.

-> Power flow calculation
            Numba: Off
   Sparse solver: KLU
 Solution method: NR method
Power flow initialized in 0.0033 seconds.
0: |F(x)| = 0.5605182134
1: |F(x)| = 0.006202200332
2: |F(x)| = 0.1543556844
3: |F(x)| = 0.05142223718
4: |F(x)| = 0.0005148845073
5: |F(x)| = 5.687884333e-08
Converged in 6 iterations in 0.0033 seconds.
Report saved to "ieee14_out.txt" in 0.0006 seconds.

-> Continuation Power Flow
Parameterization: pseudo_arclength
       Step size: 0.1
   Adaptive step: On
         Stop at: NOSE
Nose detected at step 7, lambda=0.722794

-> CPF Report
       Converged: True
            Steps: 8
       Max lambda: 0.753555
      Termination: Nose point at lambda=0.753555
CPF completed in 8 steps in 0.0169 seconds. max lambda = 0.753555
```

```
weak_noq = np.argmax(ss_noq.CPF.V[:, 0] - ss_noq.CPF.V[:, np.argmax(ss_noq.
↪CPF.lam)])

fig, ax = plt.subplots()
ax.plot(ss_noq.CPF.lam, ss_noq.CPF.V[weak_noq, :], 'k-', linewidth=1.2, label=
↪'No Q limits')
ax.plot(ss_qlim.CPF.lam, ss_qlim.CPF.V[weak_noq, :], 'r--', linewidth=1.2,␣
↪label='With Q limits')
ax.set_xlabel(r'$\lambda$')
ax.set_ylabel(f'|V| at Bus {ss_noq.Bus.idx.v[weak_noq]} (pu)')
ax.legend()
ax.grid(True, alpha=0.3)
fig.tight_layout()
plt.show()
```



2.14.11 Cleanup

```
!andes misc -C
```

```
"/home/docs/checkouts/readthedocs.org/user_builds/andes/checkouts/stable/docs/
↪source/tutorials/ieee14_out.txt" removed.
```

2.14.12 Next Steps

- *State Estimation*: State estimation with WLS and custom measurements
- *Eigenvalue Analysis*: Small-signal stability assessment

2.15 Inspecting Model Equations

One of the key features of ANDES is its symbolic-numeric hybrid framework, which allows you to inspect the mathematical equations that define each model. This capability is invaluable for understanding model behavior, debugging simulations, and verifying that models are implemented correctly.

Every model in ANDES is defined symbolically using SymPy, and the framework automatically generates numerical code for efficient simulation. This tutorial shows how to access and examine the symbolic equations, variables, and services that comprise each model.

2.15.1 Setup

```
%matplotlib inline

import andes

andes.config_logger(30)  # Reduce logging verbosity
```

2.15.2 Listing Available Models

To see all models supported by ANDES, create an empty System object and call `supported_models()`. Models are organized into groups based on their function in the power system.

```
ss = andes.System()
print(ss.supported_models())
```

```
Supported Groups and Models

      Group         |                           Models
-------------------+-------------------------------------------------------------
 ACLine             | Line
```

(continues on next page)

(continued from previous page)

```
ACNode             | Bus
ACShort            | Jumper
Calculation        | ACE, ACEc, COI
Collection         | Area
DCLink             | Ground, R, L, C, RCp, RCs, RLs, RLCs, RLCp
DCTopology         | Node
DG                 | PVD1, ESD1, EV1, EV2
DGProtection       | DGPRCT1, DGPRCTExt
DataSeries         | TimeSeries
DynLoad            | ZIP, FLoad
DynShaft           | SHAFT5
Exciter            | EXDC2, IEEEX1, ESDC1A, ESDC2A, EXST1, ESST3A, SEXS,
                   | IEEET1, EXAC1, EXAC2, EXAC4, ESST4B, AC8B, IEEET3,
                   | ESAC1A, ESST1A, ESAC5A
FreqMeasurement    | BusFreq, BusROCOF, FreqDiv
Information        | Summary
Interface          | Fortescue
Motor              | Motor3, Motor5
OutputSelect       | Output
PLL                | PLL1, PLL2
PSS                | IEEEST, ST2CUT
PhasorMeasurement  | PMU
RenAerodynamics    | WTARA1, WTARV1
RenExciter         | REECA1, REECA1E, REECA1G, REECB1
RenGen             | REGCA1, REGCP1, REGCV1, REGCV2, REGF1, REGF2, REGF3
RenGovernor        | WTDTA1, WTDS
RenPitch           | WTPTA1
RenPlant           | REPCA1
RenTorque          | WTTQA1
StaticACDC         | VSCShunt
StaticGen          | PV, Slack
StaticLoad         | PQ
StaticShunt        | Shunt, ShuntTD, ShuntSw
SynGen             | GENCLS, GENROU, PLBVFU1
TimedEvent         | Toggle, Fault, Alter
TurbineGov         | TG2, TGOV1, TGOV1DB, TGOV1N, TGOV1NDB, IEEEG1, IEESGO,
                   | GAST, HYGOV, HYGOVDB, HYGOV4
VoltComp           | IEEEVC
```

2.15.3 Preparing Model Equations

Before inspecting a model's symbolic equations, you need to call `prepare()` on that model. This triggers the symbolic processing that generates the equations and Jacobian matrices. For a single model, this is fast; preparing all models takes longer but is still manageable.

```
# Prepare equations for the GENCLS model (classical generator)
ss.GENCLS.prepare()
```

2.15.4 Model Documentation

Each model provides comprehensive documentation through the `doc()` method. This returns a formatted string containing:

- Model description and group membership
- Parameter definitions with units and defaults
- Variable definitions (states and algebraics)
- Initialization equations
- Differential and algebraic equations
- Service definitions
- Configuration options

```
print(ss.GENCLS.doc())
```

```
Model <GENCLS> in Group <SynGen>
Classical generator model.

Parameters

 Name    |        Description         | Default | Unit |         Properties
--------+---------------------------+---------+------+-------------------------
 idx     | unique device idx          |         |      |
 u       | connection status          | 1       | bool |
 name    | device name                |         |      |
 bus     | interface bus id           |         |      | mandatory
 gen     | static generator index     |         |      | mandatory
 coi     | center of inertia index    |         |      |
 coi2    | center of inertia index    |         |      |
 Sn      | Power rating               | 100     | MVA  |
 Vn      | AC voltage rating          | 110     |      |
 fn      | rated frequency            | 60      |      |
 D       | Damping coefficient        | 0       |      | power
 M       | machine start up time      | 6       |      | non_zero,non_negative,po
         | (2H)                       |         |      | wer
 ra      | armature resistance        | 0       |      | z
 xl      | leakage reactance          | 0       |      | z
 xd1     | d-axis transient           | 0.302   |      | z
         | reactance                  |         |      |
 kp      | active power feedback      | 0       |      |
         | gain                       |         |      |
 kw      | speed feedback gain        | 0       |      |
 S10     | first saturation factor    | 0       |      |
 S12     | second saturation factor   | 1       |      |
 gammap  | P ratio of linked static   | 1       |      |
         | gen                        |         |      |
 gammaq  | Q ratio of linked static   | 1       |      |
         | gen                        |         |      |
```

(continues on next page)

(continued from previous page)

```
 ue     | effective online status  | 0       |      |
 subidx | Generator idx in plant;  | 0       |      |
        | only used by PSS/E data  |         |      |

Variables

 Name   |   Type   |            Description            |   Unit   | Properties
--------+----------+-----------------------------------+----------+-----------
 delta  | State    | rotor angle                       | rad      | v_str
 omega  | State    | rotor speed                       | pu (Hz)  | v_str
 Id     | Algeb    | d-axis current                    |          | v_str
 Iq     | Algeb    | q-axis current                    |          | v_str
 vd     | Algeb    | d-axis voltage                    |          | v_str
 vq     | Algeb    | q-axis voltage                    |          | v_str
 tm     | Algeb    | mechanical torque                 |          | v_str
 te     | Algeb    | electric torque                   |          | v_str
 vf     | Algeb    | excitation voltage                | pu       | v_str
 XadIfd | Algeb    | d-axis armature excitation        | p.u (kV) | v_str
        |          | current                           |          |
 Pe     | Algeb    | active power injection            |          | v_str
 Qe     | Algeb    | reactive power injection          |          | v_str
 psid   | Algeb    | d-axis flux                       |          | v_str
 psiq   | Algeb    | q-axis flux                       |          | v_str
 a      | ExtAlgeb | Bus voltage phase angle           |          |
 v      | ExtAlgeb | Bus voltage magnitude             |          |

Initialization Equations

 Name   |   Type   |        Initial Value
--------+----------+----------------------------
 delta  | State    | delta0
 omega  | State    | u
 Id     | Algeb    | u * Id0
 Iq     | Algeb    | u * Iq0
 vd     | Algeb    | u * vd0
 vq     | Algeb    | u * vq0
 tm     | Algeb    | tm0
 te     | Algeb    | u * tm0
 vf     | Algeb    | u * vf0
 XadIfd | Algeb    | u * vf0
 Pe     | Algeb    | u * (vd0 * Id0 + vq0 * Iq0)
 Qe     | Algeb    | u * (vq0 * Id0 - vd0 * Iq0)
 psid   | Algeb    | u * psid0
 psiq   | Algeb    | u * psiq0
 a      | ExtAlgeb |
 v      | ExtAlgeb |

Differential Equations

 Name  | Type  | RHS of Equation "T x' = f(x, y)" | T (LHS)
-------+-------+-----------------------------------+--------
 delta | State | ue * (2 * pi * fn) * (omega - 1) |
```

(continues on next page)

(continued from previous page)

```
 omega | State | ue * (tm - te - D * (omega - 1)) | M

Algebraic Equations

 Name   |   Type   |   RHS of Equation "0 = g(x, y)"
--------+----------+----------------------------------
 Id     | Algeb    | + xq * Id - vf+ psid
 Iq     | Algeb    | + xq * Iq+ psiq
 vd     | Algeb    | ue * v * sin(delta - a) - vd
 vq     | Algeb    | ue * v * cos(delta - a) - vq
 tm     | Algeb    | tm0 - tm
 te     | Algeb    | ue * (psid * Iq - psiq * Id) - te
 vf     | Algeb    | ue * vf0 - vf
 XadIfd | Algeb    | ue * vf0 - XadIfd
 Pe     | Algeb    | ue * (vd * Id + vq * Iq) - Pe
 Qe     | Algeb    | ue * (vq * Id - vd * Iq) - Qe
 psid   | Algeb    | ue * (ra*Iq + vq) - psid
 psiq   | Algeb    | ue * (ra*Id + vd) + psiq
 a      | ExtAlgeb | -ue * (vd * Id + vq * Iq)
 v      | ExtAlgeb | -ue * (vq * Id - vd * Iq)

Services

  Name   |                       Equation                       |     Type
---------+------------------------------------------------------+-------------
 p0      | p0s * gammap                                         | ConstService
 q0      | q0s * gammaq                                         | ConstService
 _V      | v * exp(1j * a)                                      | ConstService
 _S      | p0 - 1j * q0                                         | ConstService
 _I      | _S / conj(_V)                                        | ConstService
 _E      | _V + _I * (ra + 1j * xq)                             | ConstService
 _deltac | log(_E / abs(_E))                                    | ConstService
 delta0  | u * im(_deltac)                                      | ConstService
 vdq     | u * (_V * exp(1j * 0.5 * pi - _deltac))              | ConstService
 Idq     | u * (_I * exp(1j * 0.5 * pi - _deltac))              | ConstService
 Id0     | re(Idq)                                              | ConstService
 Iq0     | im(Idq)                                              | ConstService
 vd0     | re(vdq)                                              | ConstService
 vq0     | im(vdq)                                              | ConstService
 tm0     | u * ((vq0 + ra * Iq0) * Iq0 + (vd0 + ra * Id0) *     | ConstService
         | Id0)                                                 |
 psid0   | u * (ra * Iq0) + vq0                                 | ConstService
 psiq0   | -u * (ra * Id0) - vd0                                | ConstService
 vf0     | (vq0 + ra * Iq0) + xq * Id0                          | ConstService

Setpoints

Name  | Target | Info
------+--------+-----
 pref | tm0    |
 vref | vf0    |
```

(continues on next page)

(continued from previous page)

```
Config Fields in [GENCLS]

   Option      | Value |                 Info                 | Acceptable values
---------------+-------+--------------------------------------+------------------
 allow_adjust  | 1     | allow adjusting upper or lower       | (0, 1)
               |       | limits                               |
 adjust_lower  | 0     | adjust lower limit                   | (0, 1)
 adjust_upper  | 1     | adjust upper limit                   | (0, 1)
 vf_lower      | 1     | lower limit for vf warning           |
 vf_upper      | 5     | upper limit for vf warning           |
```

The documentation shows that GENCLS (classical generator) has two state variables (`delta` for rotor angle and `omega` for rotor speed) and several algebraic variables for currents, voltages, and power outputs. The differential equations implement the swing equation that governs generator dynamics.

2.15.5 Accessing Variables

Model variables are stored in ordered dictionaries accessible through `states` and `algebs` attributes. This lets you enumerate all variables programmatically.

```python
# State (differential) variables
print("State variables:")
for name, var in ss.GENCLS.states.items():
    print(f"  {name}: {var}")
```

```
State variables:
  delta: State: GENCLS.delta, []
  omega: State: GENCLS.omega, []
```

```python
# Algebraic variables
print("Algebraic variables:")
for name, var in ss.GENCLS.algebs.items():
    print(f"  {name}: {var}")
```

```
Algebraic variables:
  Id: Algeb: GENCLS.Id, []
  Iq: Algeb: GENCLS.Iq, []
  vd: Algeb: GENCLS.vd, []
  vq: Algeb: GENCLS.vq, []
  tm: Algeb: GENCLS.tm, []
  te: Algeb: GENCLS.te, []
  vf: Algeb: GENCLS.vf, []
  XadIfd: Algeb: GENCLS.XadIfd, []
  Pe: Algeb: GENCLS.Pe, []
  Qe: Algeb: GENCLS.Qe, []
  psid: Algeb: GENCLS.psid, []
  psiq: Algeb: GENCLS.psiq, []
```

2.15.6 Symbolic Equations

The symbolic equations are stored in `Model.syms`. This object contains SymPy expressions that can be displayed, manipulated, and analyzed. The main attributes are:

Attribute	Contents
`xy`	Vector of all variables (states then algebraics)
`f`	Differential equations (right-hand side of T x' = f)
`g`	Algebraic equations (right-hand side of 0 = g)
`df`	Jacobian df/dxy
`dg`	Jacobian dg/dxy
`s`	Service equations

```
# All variables in order
ss.GENCLS.syms.xy
```

$$\begin{bmatrix} \delta \\ \omega \\ Id \\ Iq \\ vd \\ vq \\ tm \\ te \\ vf \\ XadIfd \\ Pe \\ Qe \\ psid \\ psiq \\ a \\ v \end{bmatrix}$$

Differential Equations

The differential equations define how state variables evolve over time. For generators, these typically include the swing equation relating rotor angle and speed to mechanical and electrical torque.

```
ss.GENCLS.syms.f
```

$$\begin{bmatrix} 2\pi fnue\left(\omega - 1\right) \\ ue\left(-D\left(\omega - 1\right) - te + tm\right) \end{bmatrix}$$

The first equation is the rotor angle dynamics: `d(delta)/dt = 2*pi*fn*(omega - 1)`, where `omega` is in per-unit of nominal frequency. The second equation is the swing equation: `M*d(omega)/dt = tm - te - D*(omega - 1)`, relating acceleration to the imbalance between mechanical torque `tm` and electrical torque `te`, with damping `D`.

Algebraic Equations

Algebraic equations define relationships that must hold at every instant (they have no time derivative). These typically include current-voltage relationships, power calculations, and interface equations.

```
ss.GENCLS.syms.g
```

$$\left[\begin{array}{c} Idxq + psid - vf \\ Iqxq + psiq \\ -uev\sin\left(a - \delta\right) - vd \\ uev\cos\left(a - \delta\right) - vq \\ -tm + tm_0 \\ -te + ue\left(-Idpsiq + Iqpsid\right) \\ uevf_0 - vf \\ -XadIfd + uevf_0 \\ -Pe + ue\left(Idvd + Iqvq\right) \\ -Qe + ue\left(Idvq - Iqvd\right) \\ -psid + ue\left(Iqra + vq\right) \\ psiq + ue\left(Idra + vd\right) \\ -ue\left(Idvd + Iqvq\right) \\ -ue\left(Idvq - Iqvd\right) \end{array}\right]$$

Jacobian Matrices

The Jacobian matrices contain partial derivatives of the equations with respect to variables. These are essential for Newton-Raphson iteration in power flow and implicit integration in time-domain simulation.

```
# Jacobian of differential equations
ss.GENCLS.syms.df
```

$$\left[\begin{array}{cccccccccccccccc} 0 & 2\pi fnue & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -Due & 0 & 0 & 0 & 0 & ue & -ue & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right]$$

```
# Jacobian of algebraic equations (first 4 rows, first 6 columns)
ss.GENCLS.syms.dg[:4, :6]
```

$$\begin{bmatrix} 0 & 0 & xq & 0 & 0 & 0 \\ 0 & 0 & 0 & xq & 0 & 0 \\ uev\cos(a-\delta) & 0 & 0 & 0 & -1 & 0 \\ uev\sin(a-\delta) & 0 & 0 & 0 & 0 & -1 \end{bmatrix}$$

2.15.7 Services

Services are intermediate calculations that are computed once (at initialization) and reused throughout the simulation. They often represent derived quantities like initial values or converted parameters.

```
# List all services
print("Services:")
for name, svc in ss.GENCLS.services.items():
    print(f"  {name}")
```

```
Services:
  p0
  q0
  _V
  _S
  _I
  _E
  _deltac
  delta0
  vdq
  Idq
  Id0
  Iq0
  vd0
  vq0
  tm0
  psid0
  psiq0
  vf0
```

```
# Service equations
ss.GENCLS.syms.s
```

```
[gammap*p0s,
 gammaq*q0s,
 v*exp(I*a),
 p0 - I*q0,
 _S/conj(_V),
 _I*(ra + I*xq) + _V,
 log(_E/Abs(_E)),
 u*im(_deltac),
 _V*u*exp(-_deltac + 0.5*I*pi),
```

(continues on next page)

(continued from previous page)

```
_I*u*exp(-_deltac + 0.5*I*pi),
re(Idq),
im(Idq),
re(vdq),
im(vdq),
u*(Id0*(Id0*ra + vd0) + Iq0*(Iq0*ra + vq0)),
Iq0*ra*u + vq0,
-Id0*ra*u - vd0,
Id0*xq + Iq0*ra + vq0]
```

2.15.8 Practical Applications

Understanding model equations is useful for:

- **Verification**: Confirming that a model implements the expected physics
- **Debugging**: Identifying why a simulation produces unexpected results
- **Extension**: Understanding the structure before adding new features
- **Documentation**: Generating mathematical descriptions for papers or reports
- **Teaching**: Demonstrating power system dynamics concepts

2.15.9 See Also

- *Hybrid Symbolic-Numeric Framework* - Overview of the symbolic-numeric framework
- *Variables* - Detailed variable type documentation
- *Creating Models* - Creating new models

2.16 Reinforcement Learning with ANDES

Reinforcement learning (RL) offers a data-driven approach to designing power system controllers. Rather than manually tuning control laws, an RL agent learns a control policy by interacting with the simulation environment and optimizing a user-defined objective. Applications include automatic generation control (AGC), voltage regulation, and emergency frequency response.

ANDES provides the `AndesEnv` class, a Gymnasium-compatible environment that wraps ANDES time-domain simulation. Each `step()` advances the simulation by a fixed time interval, and the agent observes system variables and applies control actions through the standard Gymnasium API. The environment uses `TDS.reinit()` for fast episode resets (~1 ms), making it practical for training algorithms that require thousands of episodes.

This tutorial demonstrates how to construct an RL environment, define observations, actions, and reward functions, and run training episodes. Familiarity with the setpoint API from *Dynamic Control and Setpoint Changes* and the frequency response concepts from *Frequency Response and Load Shedding* is assumed.

Note: Prerequisites:

- Complete *Dynamic Control and Setpoint Changes* for the setpoint API and multi-stage simulation.
- Complete *Frequency Response and Load Shedding* for frequency dynamics after disturbances.
- Install the RL extra: `pip install andes[rl]`

```
# Reduce logging verbosity for PDF builds
import os
if os.environ.get('SPHINX_BUILD_PDF'):
    import andes
    _orig_config_logger = andes.config_logger
    def _quiet_logger(stream_level=20, *args, **kwargs):
        stream_level = max(stream_level, 30)
        return _orig_config_logger(stream_level, *args, **kwargs)
    andes.config_logger = _quiet_logger
```

2.16.1 Setup

```
import numpy as np

from andes.rl import AndesEnv
from andes.utils.paths import get_case
```

2.16.2 Constructing the Environment

An `AndesEnv` instance requires four arguments: a case file, an observation specification, an action specification, and a reward function. Together, these define the RL problem: what the agent sees, what it can do, and what it optimizes.

The following example constructs an environment for secondary frequency control (AGC) on the IEEE 14-bus system. The agent observes generator rotor speeds and controls the auxiliary power input of all synchronous generators.

```
CASE = get_case('ieee14/ieee14_esst3a.xlsx')


def agc_reward(obs, action, env):
    """Penalize frequency deviation and control effort."""
    freq_dev = obs - 1.0
    return -float(np.sum(freq_dev ** 2) + 0.01 * np.sum(action ** 2))


env = AndesEnv(
    case=CASE,
    obs=[('GENROU', 'omega')],
    acts=[('SynGen', 'paux')],
```

(continues on next page)

(continued from previous page)

```
    reward_fn=agc_reward,
    dt=0.1,
    tf=5.0,
)
```

The constructor performs several initialization steps internally: it loads the case, runs power flow, initializes TDS, and resolves the observation and action addresses. If any specification is invalid (e.g., a nonexistent model or setpoint), a `ValueError` is raised immediately.

The resulting spaces can be inspected:

```
print(f"Observation space: {env.observation_space}")
print(f"Action space:      {env.action_space}")
print(f"Number of generators observed: {env.observation_space.shape[0]}")
print(f"Number of generators controlled: {env.action_space.shape[0]}")
```

```
Observation space: Box(-inf, inf, (5,), float32)
Action space:      Box(-inf, inf, (5,), float32)
Number of generators observed: 5
Number of generators controlled: 5
```

Observation Specification

The `obs` parameter is a list of tuples specifying which variables the agent observes. Each tuple takes the form `(model, variable)` or `(model, variable, idx_list)`. The variable type (state or algebraic) is resolved automatically from the model definition.

Spec	Observation
`('GENROU', 'omega')`	All GENROU rotor speeds
`('Bus', 'v')`	All bus voltage magnitudes
`('Bus', 'v', [1, 5, 14])`	Voltages at buses 1, 5, and 14 only

Multiple specs are concatenated into a single flat observation vector. For example, observing both rotor speeds and selected bus voltages:

```
env_multi = AndesEnv(
    case=CASE,
    obs=[
        ('GENROU', 'omega'),          # all generator speeds
        ('Bus', 'v', [1, 2, 3]),      # voltages at 3 buses
    ],
    acts=[('SynGen', 'paux')],
    reward_fn=agc_reward,
    dt=0.1, tf=5.0,
)

n_gen = env_multi.ss.GENROU.n
```

(continues on next page)

(continued from previous page)

```
print(f"Obs dimension: {env_multi.observation_space.shape[0]} "
      f"({n_gen} speeds + 3 voltages)")
```

```
Obs dimension: 8 (5 speeds + 3 voltages)
```

Action Specification

The `acts` parameter is a list of `(target, setpoint)` tuples. The target can be either a **group name** or a **model name**, and the behavior differs accordingly:

- **Group target** (e.g., `'SynGen'`): Uses the group-level setpoint API (`set_paux`, `set_pref`, `set_vref`). This resolves the controller chain automatically, as described in *Dynamic Control and Setpoint Changes*. One action dimension is created per device in the group.
- **Model target** (e.g., `'TGOV1'`): Writes directly to the specified attribute on the model. One action dimension is created per device in the model.

The group-level approach is recommended for most applications because it does not require knowledge of which specific governor or exciter model is in use.

Tip: To bound the action space, pass `action_low` and `action_high` to the constructor. For RL algorithms that assume normalized actions, symmetric bounds such as `[-0.1, 0.1]` are typical.

Reward Function

The reward function defines the optimization objective. It receives three arguments: the observation vector, the action vector, and the environment instance. It must return a scalar float.

The `agc_reward` function used above penalizes two quantities: the squared frequency deviation from nominal (1.0 pu) and the squared control effort (weighted by 0.01). This structure encourages the agent to restore frequency with minimal actuator usage.

Other reward designs are possible depending on the application:

```
# Voltage regulation: penalize voltage deviation from setpoint
def voltage_reward(obs, action, env):
    v_dev = obs - 1.0    # assuming obs contains bus voltages
    return -float(np.sum(v_dev ** 2))

# Frequency nadir: penalize the worst-case deviation
def nadir_reward(obs, action, env):
    return -float(np.max(np.abs(obs - 1.0)))
```

The environment instance is passed as the third argument, providing access to the underlying ANDES system via `env.ss`. This allows the reward function to query any system quantity, not only the observed variables.

2.16.3 Running an Episode

The environment follows the standard Gymnasium API. An episode begins with `reset()`, which calls `TDS.reinit()` internally, and proceeds through a sequence of `step()` calls. Each step advances the simulation by `dt` seconds and returns the standard five-tuple `(obs, reward, terminated, truncated, info)`.

```
obs, info = env.reset(seed=42)
print(f"Initial obs (omega): {obs}")
print(f"Initial time: {info['t']}")
```

```
Initial obs (omega): [1. 1. 1. 1. 1.]
Initial time: 0.0
```

```
# Run a complete episode with zero actions (no control)
obs, info = env.reset(seed=42)
total_reward = 0.0
obs_history = [obs.copy()]

while True:
    action = np.zeros(env.action_space.shape)
    obs, reward, terminated, truncated, info = env.step(action)
    total_reward += reward
    obs_history.append(obs.copy())
    if terminated or truncated:
        break

print(f"Episode finished: steps={info['step']}, t={info['t']:.1f}s")
print(f"Total reward: {total_reward:.4f}")
print(f"Final omega: {obs}")
```

```
Episode finished: steps=50, t=5.0s
Total reward: -0.0000
Final omega: [1.0000004  0.9999995  0.99999994 0.99999976 0.9999997 ]
```

The episode terminates in two ways:

- `truncated=True`: the simulation reached `tf` (normal episode end).
- `terminated=True`: the simulation failed to converge (the agent destabilized the system).

Since no disturbance was applied and no control actions were taken, the system remained at steady state throughout the episode. The reward is approximately zero because the frequency stayed at nominal.

2.16.4 Adding Disturbances

A meaningful RL problem requires a disturbance for the agent to respond to. The `disturbance_fn` parameter accepts a callable that is invoked after each `reset()`. This function can run the simulation forward through a pre-defined event before handing control to the agent.

The IEEE 14-bus test case contains built-in Toggle events (a line trip at t=1.0s and reconnection at t=1.1s). The following disturbance function advances the simulation past these events, so the agent begins controlling at t=1.2s with the system in a post-disturbance transient state.

```
def fast_forward_past_event(env):
    """Advance past the line trip/reclose before agent takes control."""
    env.ss.TDS.config.tf = 1.2
    env.ss.TDS.run(no_summary=True)


env_dist = AndesEnv(
    case=CASE,
    obs=[('GENROU', 'omega')],
    acts=[('SynGen', 'paux')],
    reward_fn=agc_reward,
    dt=0.1,
    tf=5.0,
    disturbance_fn=fast_forward_past_event,
    action_low=-0.1,
    action_high=0.1,
)
```

```
# Run baseline episode (no control) to see the uncontrolled response
obs, info = env_dist.reset(seed=42)
baseline_obs = [obs.copy()]
baseline_reward = 0.0

while True:
    action = np.zeros(env_dist.action_space.shape)
    obs, reward, terminated, truncated, info = env_dist.step(action)
    baseline_obs.append(obs.copy())
    baseline_reward += reward
    if terminated or truncated:
        break

baseline_obs = np.array(baseline_obs)
print(f"Baseline total reward: {baseline_reward:.4f}")
print(f"Max frequency deviation: {np.max(np.abs(baseline_obs - 1.0)):.6f} pu")
```

```
Baseline total reward: -0.0000
Max frequency deviation: 0.000827 pu
```

The nonzero reward and frequency deviation confirm that the disturbance creates a meaningful control problem for the agent to solve.

2.16.5 Comparing Control Policies

Before training an RL agent, it is instructive to compare the uncontrolled baseline against a simple hand-crafted policy. The following proportional controller applies a corrective action proportional to the mean frequency deviation, analogous to a basic droop response.

```
def proportional_policy(obs, gain=5.0):
    """Simple proportional controller: action = -gain * mean(omega - 1)."""
    mean_dev = np.mean(obs - 1.0)
    action = np.full(env_dist.action_space.shape, -gain * mean_dev)
    return np.clip(action, -0.1, 0.1)


obs, info = env_dist.reset(seed=42)
controlled_obs = [obs.copy()]
controlled_reward = 0.0

while True:
    action = proportional_policy(obs)
    obs, reward, terminated, truncated, info = env_dist.step(action)
    controlled_obs.append(obs.copy())
    controlled_reward += reward
    if terminated or truncated:
        break

controlled_obs = np.array(controlled_obs)
print(f"Controlled total reward: {controlled_reward:.4f}")
print(f"Max frequency deviation: {np.max(np.abs(controlled_obs - 1.0)):.6f} pu
↪")
print(f"Improvement over baseline: {controlled_reward - baseline_reward:.4f}")
```

```
Controlled total reward: -0.0000
Max frequency deviation: 0.000855 pu
Improvement over baseline: -0.0000
```

The proportional policy should yield a higher (less negative) total reward than the uncontrolled baseline. An RL agent trained with a suitable algorithm can potentially discover a more effective policy by jointly optimizing the control actions across all generators and time steps.

2.16.6 Deterministic Reset

The `reset()` method uses `TDS.reinit()` internally, which restores the system to its exact post-initialization state. This makes episodes fully deterministic: identical seeds and actions produce identical trajectories. This property is important for reproducible training and for debugging reward functions.

```
# Verify determinism: two identical episodes should produce the same␣
↪trajectory
def run_episode(env, seed=42):
    obs_list = []
    obs, _ = env.reset(seed=seed)
```

(continues on next page)

(continued from previous page)

```
    obs_list.append(obs.copy())
    for _ in range(5):
        action = np.zeros(env.action_space.shape)
        obs, _, term, trunc, _ = env.step(action)
        obs_list.append(obs.copy())
        if term or trunc:
            break
    return np.array(obs_list)

traj1 = run_episode(env_dist)
traj2 = run_episode(env_dist)
print(f"Trajectories identical: {np.allclose(traj1, traj2, atol=1e-10)}")
```

```
Trajectories identical: True
```

2.16.7 Training with Stable-Baselines3

Because `AndesEnv` is Gymnasium-compatible, it can be used directly with standard RL libraries such as Stable-Baselines3. The following example trains a PPO agent for a small number of steps to demonstrate the integration. Production training typically requires 50,000 to 500,000 timesteps depending on the problem complexity.

Note: Stable-Baselines3 is not included in the `andes[rl]` extra. Install it separately with `pip install stable-baselines3`.

```
from stable_baselines3 import PPO

model = PPO(
    "MlpPolicy",
    env_dist,
    learning_rate=3e-4,
    n_steps=256,
    batch_size=64,
    n_epochs=10,
    verbose=1,
)

model.learn(total_timesteps=20_000)

# Evaluate the trained policy
obs, info = env_dist.reset(seed=99)
total_reward = 0.0
while True:
    action, _ = model.predict(obs, deterministic=True)
    obs, reward, terminated, truncated, info = env_dist.step(action)
    total_reward += reward
    if terminated or truncated:
```

(continues on next page)

(continued from previous page)

```
        break

print(f"Trained agent reward: {total_reward:.4f}")
```

A complete training script with logging callbacks is provided in `examples/rl_agc_training.py`.

2.16.8 Accessing the System Object

The underlying ANDES `System` object is accessible via `env.ss`. This provides full access to all models, parameters, and DAE arrays, which is useful for constructing reward functions that depend on quantities beyond the observation vector.

```
ss = env.ss

print(f"System frequency: {ss.config.freq} Hz")
print(f"Number of buses: {ss.Bus.n}")
print(f"GENROU devices: {list(ss.GENROU.idx.v)}")
print(f"Current sim time: {float(ss.dae.t):.2f} s")
```

```
System frequency: 60 Hz
Number of buses: 14
GENROU devices: ['GENROU_1', 'GENROU_2', 'GENROU_3', 'GENROU_4', 'GENROU_5']
Current sim time: 5.00 s
```

2.16.9 TDS Configuration

TDS solver options can be customized via the `tds_config` parameter at construction time. This is useful for selecting the integration method or adjusting convergence tolerances. Invalid keys raise a `ValueError` immediately.

```
env_custom = AndesEnv(
    case=CASE,
    obs=[('GENROU', 'omega')],
    acts=[('SynGen', 'paux')],
    reward_fn=agc_reward,
    dt=0.1, tf=5.0,
    tds_config={'tstep': 1/60},    # smaller integration step
)
print(f"Integration step: {env_custom.ss.TDS.config.tstep:.4f} s")
```

```
Integration step: 0.0167 s
```

2.16.10 Summary

The following table summarizes the key `AndesEnv` parameters:

Parameter	Description
`case`	Path to case file (use `get_case()` for stock cases)
`obs`	List of `(model, var)` or `(model, var, idx_list)` tuples
`acts`	List of `(target, setpoint)` tuples (group or model target)
`reward_fn`	`reward_fn(obs, action, env) -> float`
`dt`	Simulation time per `step()` call [seconds]
`tf`	Episode end time [seconds]
`disturbance_fn`	`disturbance_fn(env) -> None`, called after each `reset()`
`action_low`, `action_high`	Bounds for the action space
`obs_low`, `obs_high`	Bounds for the observation space
`tds_config`	Dict of TDS config overrides

2.16.11 Next Steps

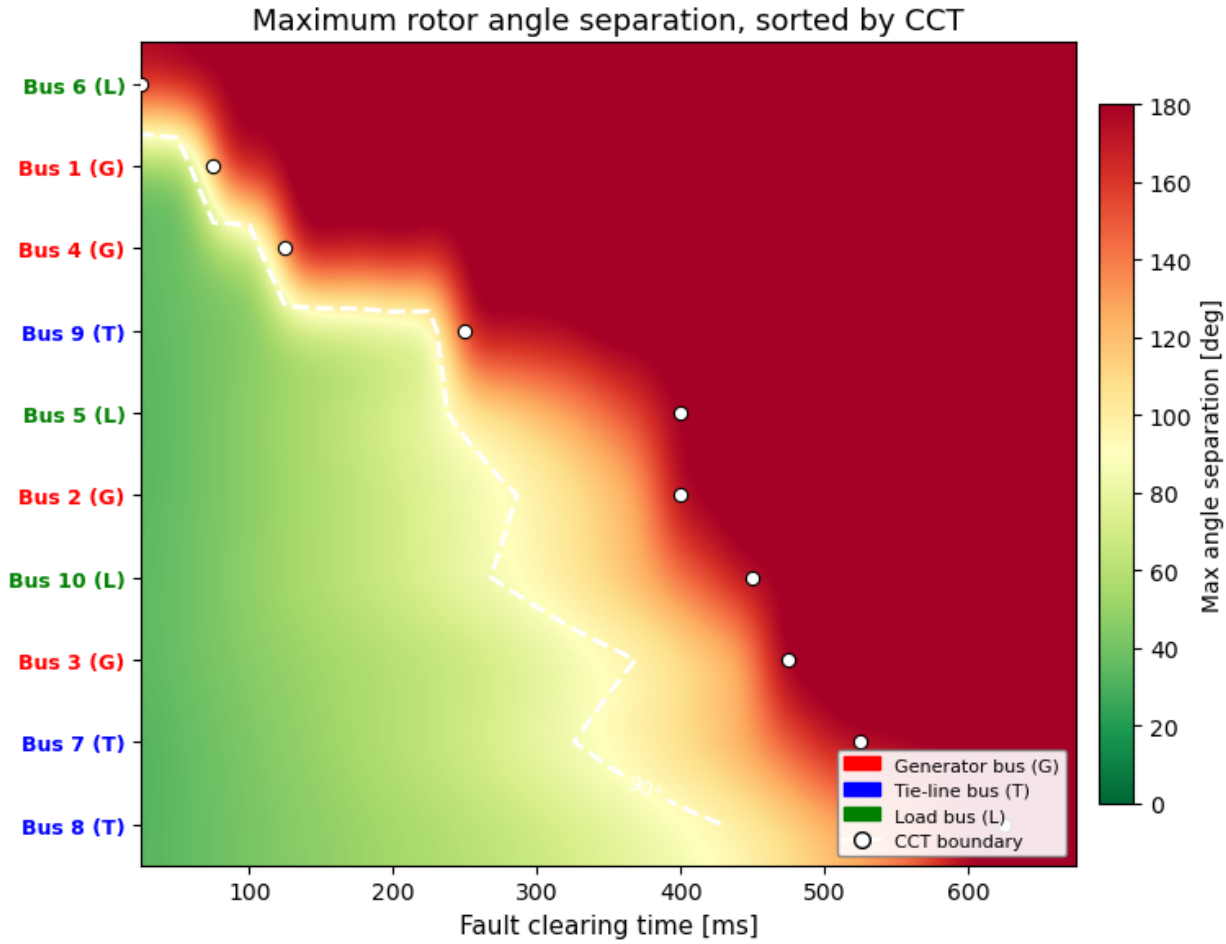
- *Dynamic Control and Setpoint Changes* for the setpoint API used by group-level actions
- *Frequency Response and Load Shedding* for frequency response fundamentals
- `examples/rl_agc_training.py` for a complete PPO training script with Stable-Baselines3

CHAPTER

THREE

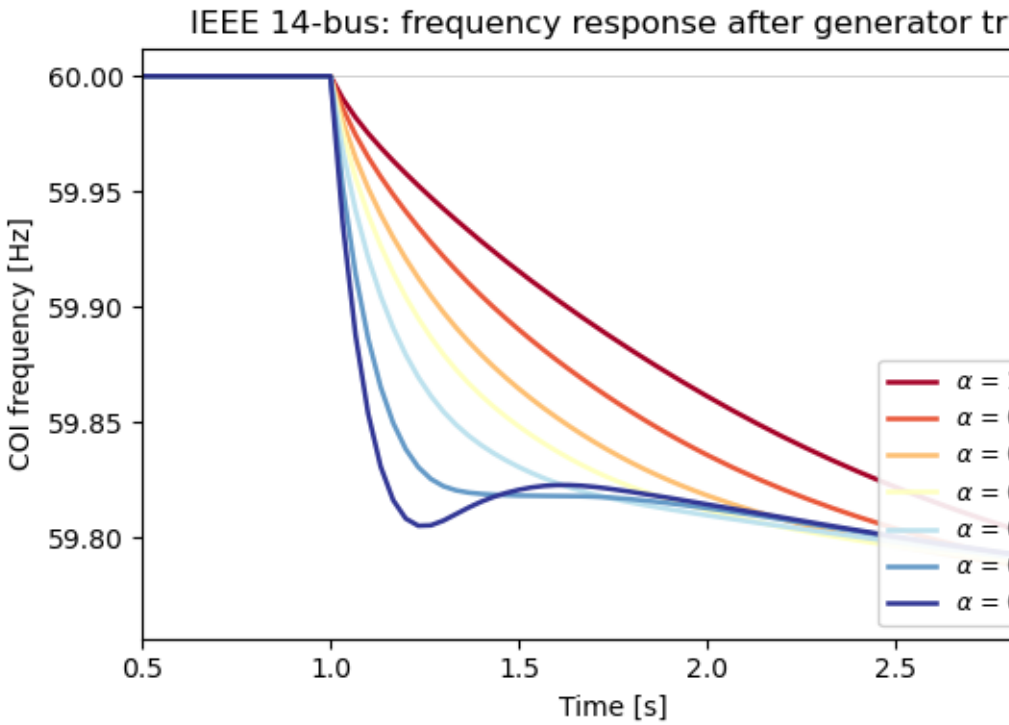
GALLERY

Application examples showcasing ANDES for power system research. Each example is a self-contained notebook that you can download and extend for your own studies.



Critical Clearing Time Mapping Sweep fault durations across all buses in the Kundur two-area system and map the transient stability boundary as a heatmap.

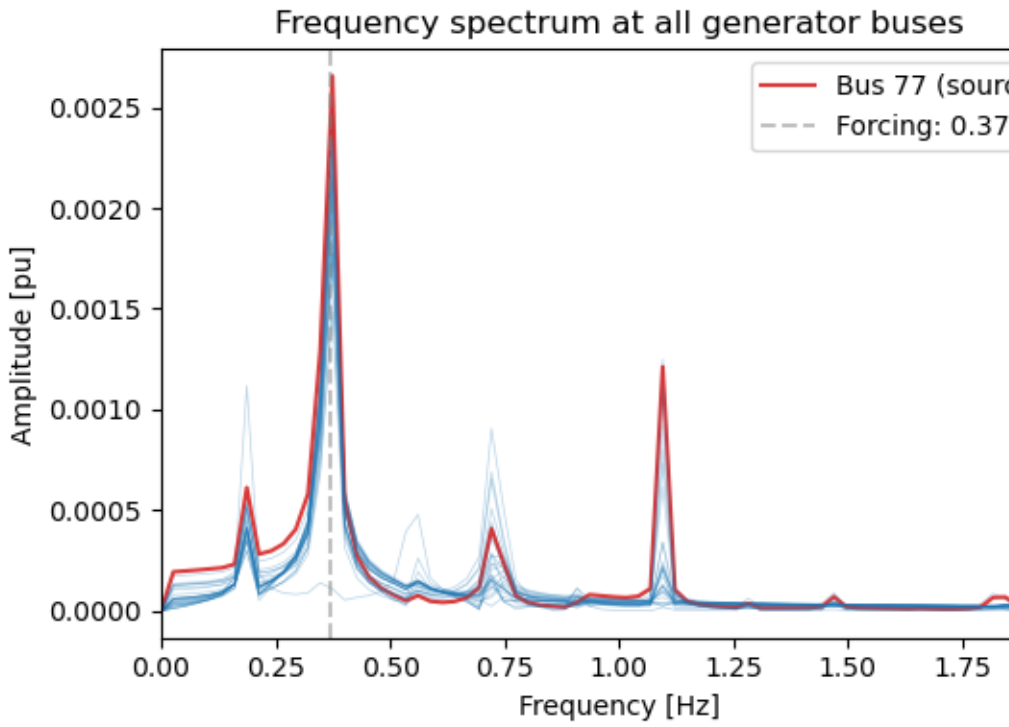
Kundur 2-area | PFlow, TDS, reinit



Critical Clearing Time Mapping

Low-Inertia Frequency Stability Scale generator inertia to emulate rising renewable penetration on the IEEE 14-bus system. Quantify how RoCoF and frequency nadir degrade.

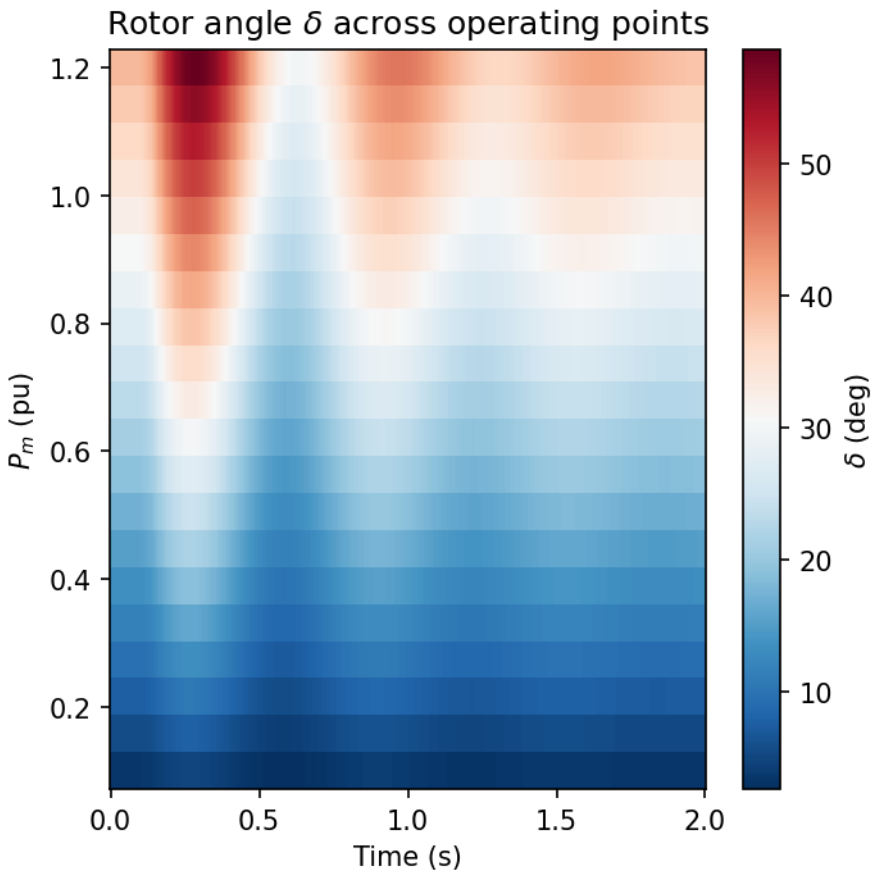
IEEE 14-bus | PFlow, TDS



Low-Inertia Frequency Stability

Forced Oscillation Source Localization Inject a periodic governor disturbance on the WECC 179-bus system and locate the source using energy flow and phase analysis from PMU data.

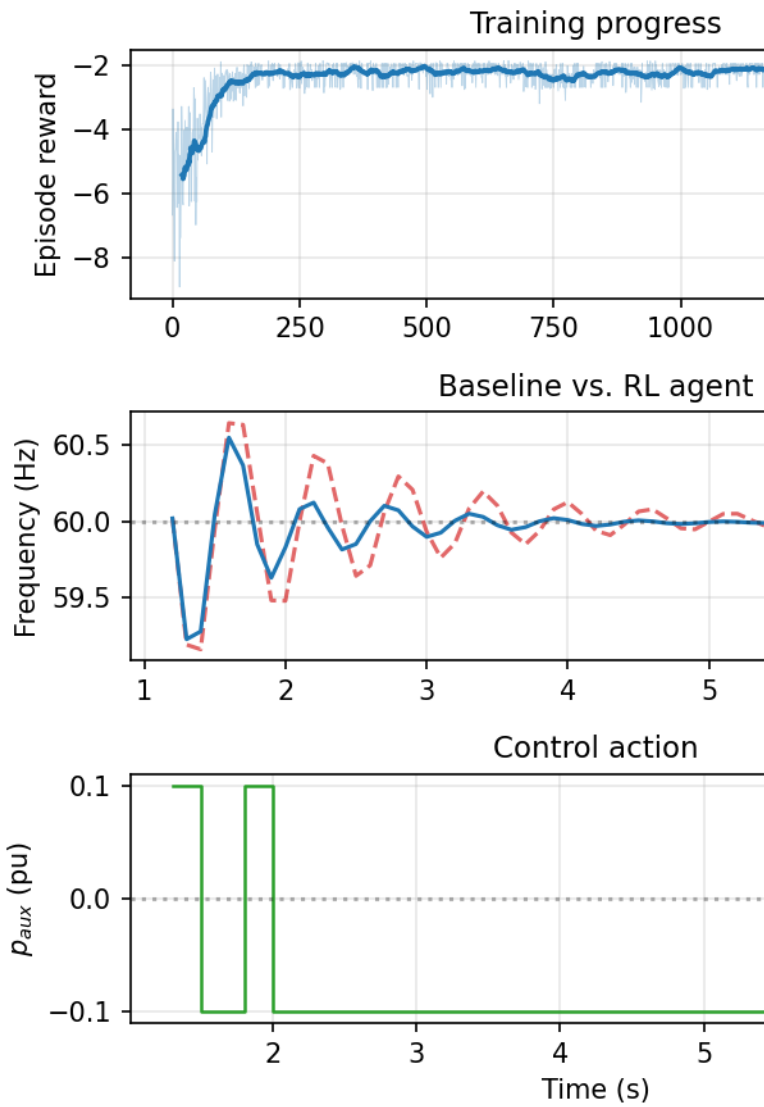
WECC 179-bus | PFlow, TDS, FFT



Forced Oscillation Source Localization

Physics-Informed Surrogate (PINN) Extract symbolic DAE equations from ANDES, eliminate algebraic variables, and train a parameterized PINN surrogate. No hand-coded physics in PyTorch.

SMIB | SymPy, PyTorch, PINN



Physics-Informed Surrogate from Symbolic Equations

RL Oscillation Damping on SMIB Train a PPO agent to damp post-fault rotor oscillations using the Gymnasium-compatible AndesEnv. Zero domain knowledge, pure simulation experience.

SMIB | PFlow, TDS, AndesEnv, PPO

Post-Fault Rotor Oscillation on an SMIB System

3.1 Running the Examples

All examples use built-in test cases and can be run directly after installing ANDES:

```
pip install andes[dev]     # includes matplotlib, pandas
pip install scikit-learn   # optional, for ML extensions
```

Each notebook is designed as a **template**: after following the walkthrough, look for the "Extend This Example" section at the end for ideas on adapting it to your own research.

3.1.1 Critical Clearing Time Mapping

The critical clearing time (CCT) is defined as the maximum duration for which a fault can persist before the system loses synchronism. It is a fundamental metric in transient stability assessment, where a shorter CCT indicates a more vulnerable operating condition. By mapping the CCT across all buses in a network, the spatial distribution of stability margins can be revealed.

In this notebook, the CCT is computed for every bus in the Kundur two-area system, which consists of four GENROU generators equipped with AVR and governor controls. The procedure is as follows:

1. A three-phase fault is applied at each bus, and the clearing time is swept over a range of values.
2. Each scenario is classified as stable, unstable, or non-convergent using the built-in rotor angle separation criterion.
3. The stability boundary is visualized through a heatmap of maximum angle separation, a network topology diagram colored by CCT, and rotor angle trajectories at the stability boundary.

The `TDS.reinit()` API is utilized to reset the simulation to its post-initialization state between scenarios, thereby avoiding the overhead of reloading the system for each clearing time.

```
%matplotlib inline

import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from matplotlib.colors import Normalize
from matplotlib.cm import ScalarMappable
import matplotlib.patches as mpatches

import andes

andes.config_logger(stream_level=50)
```

System Setup

The Kundur two-area system consists of two symmetrical areas connected by a weak tie line, with two GENROU generators in each area. The built-in case includes a line-trip event (`Toggle`) at $t = 2$ s, which is disabled here so that the fault remains the sole disturbance.

Each bus is classified by its role in the network: generator buses (G) host synchronous machines, tie-line buses (T) form the inter-area corridor, and load buses (L) represent intermediate connection points. This classification is used to annotate the visualizations that follow.

```
case_path = andes.get_case("kundur/kundur_full.xlsx")
ss_ref = andes.load(case_path, no_output=True)
ss_ref.Toggle.u.v[0] = 0  # disable built-in line trip
ss_ref.PFlow.run()

bus_idxs = list(ss_ref.Bus.idx.v)
gen_idxs = list(ss_ref.GENROU.idx.v)
```

(continues on next page)

(continued from previous page)

```
gen_buses = list(ss_ref.GENROU.bus.v)

# Classify buses by role
bus_type = {}
for b in bus_idxs:
    if b in gen_buses:
        bus_type[b] = "gen"
    elif b in [7, 8, 9]:  # tie-line buses
        bus_type[b] = "tie"
    else:
        bus_type[b] = "load"

gen_info = pd.DataFrame({
    "gen_idx": gen_idxs,
    "bus": gen_buses,
    "M": ss_ref.GENROU.M.v,
})
gen_info
```

```
   gen_idx  bus       M
0        1    1  117.00
1        2    2  117.00
2        3    3  111.15
3        4    4  111.15
```

CCT Sweep

A three-phase fault with reactance $x_f = 10^{-4}$ p.u. is applied at $t = 1$ s on each bus. The clearing time is swept from 25 ms to 675 ms in 25 ms increments. For each fault bus, the system is loaded once, and `TDS.reinit()` is called between clearing-time values to restore the post-initialization state without reloading.

Each scenario is classified into one of three outcomes:

- **Stable**: the simulation completes and all rotor angle separations remain below 180 degrees.
- **Unstable**: the built-in `deltadelta` criterion detects a rotor angle separation exceeding 180 degrees, and the simulation is terminated early.
- **Non-convergent**: the Newton solver fails to converge and the time step is reduced to zero.

The maximum rotor angle separation (in degrees) is recorded for each scenario to produce a continuous stability map.

```
tc_values = np.arange(0.025, 0.70, 0.025)
n_bus = len(bus_idxs)
n_tc = len(tc_values)

outcome = np.full((n_bus, n_tc), -1, dtype=int)
max_sep_deg = np.full((n_bus, n_tc), np.nan)
```

(continues on next page)

(continued from previous page)

```
for i, fbus in enumerate(bus_idxs):
    ss = andes.load(case_path, setup=False, no_output=True)
    ss.Toggle.u.v[0] = 0
    ss.add("Fault", bus=fbus, tf=1.0, tc=1.0 + tc_values[0], xf=1e-4)
    ss.setup()
    ss.config.warn_abnormal = 0

    ss.PFlow.run()
    ss.TDS.config.tf = 8.0
    ss.TDS.config.no_tqdm = 1
    ss.TDS.init()

    for j, tc in enumerate(tc_values):
        ss.TDS.reinit()
        ss.Fault.tc.v[0] = 1.0 + tc

        ss.switch_dict.clear()
        ss.store_switch_times(ss.exist.pflow_tds)

        ss.TDS.config.tf = 8.0
        ss.TDS.run(no_summary=True)

        # Extract max rotor angle separation
        delta = ss.dae.ts.x[:, ss.GENROU.delta.a]
        sep = np.rad2deg(np.max(delta, axis=1) - np.min(delta, axis=1))
        max_sep_deg[i, j] = sep.max()

        if not ss.TDS.busted:
            outcome[i, j] = 0
        elif "criteria" in ss.TDS.err_msg.lower():
            outcome[i, j] = 1
        else:
            outcome[i, j] = 2
```

Stability Boundary

The heatmap below displays the maximum rotor angle separation for each (bus, clearing time) pair. Buses are sorted by their CCT values, with the most vulnerable buses at the top. Green regions correspond to small angle excursions (stable operation), while red regions indicate separations approaching or exceeding the 180-degree threshold. The white dashed contour marks the 90-degree separation level, and white dots indicate the CCT boundary for each bus.

Bus labels are color-coded by type: generator (G, red), tie-line (T, blue), and load (L, green).

```
# Compute CCT per bus
cct = np.full(n_bus, np.nan)
for i in range(n_bus):
    fail_idx = np.where(outcome[i] > 0)[0]
    if len(fail_idx) > 0:
        cct[i] = tc_values[fail_idx[0]]
```

(continues on next page)

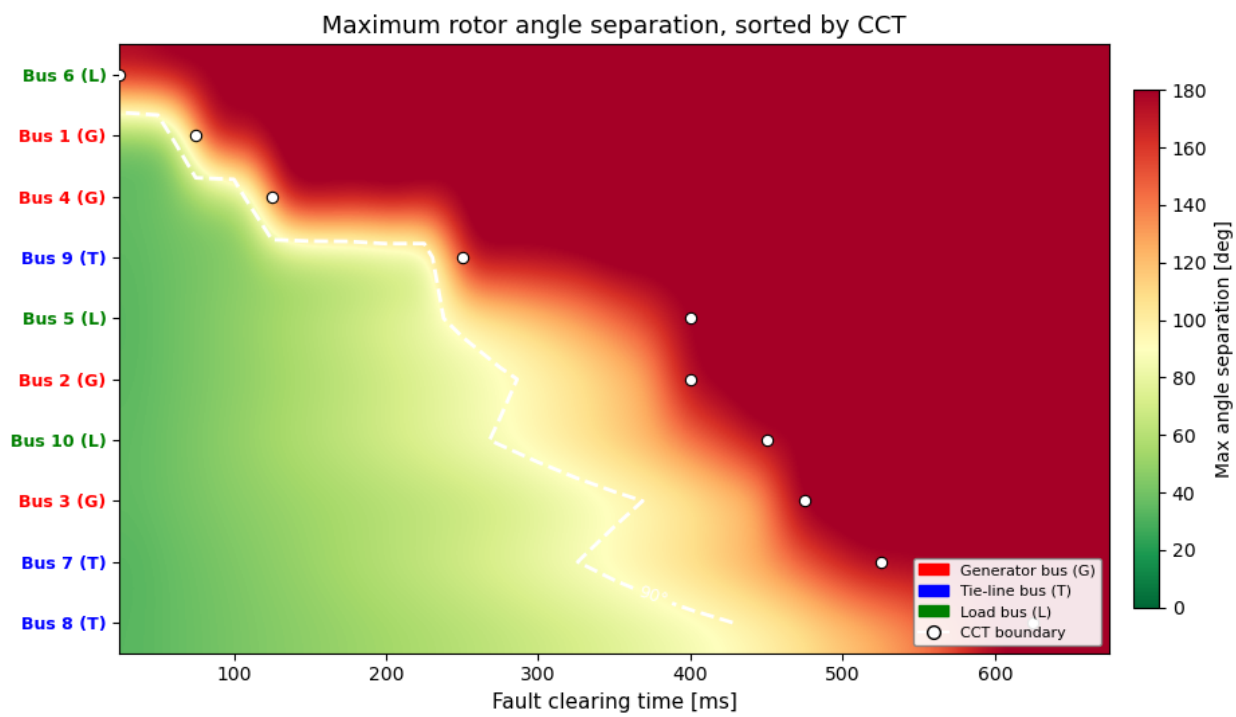
(continued from previous page)

```
    else:
        cct[i] = tc_values[-1]

# Cap non-convergent cells at threshold for smooth plotting
plot_data = max_sep_deg.copy()
plot_data[outcome == 2] = 180.0

# Sort buses by CCT (most vulnerable at top)
sort_order = np.argsort(cct)
plot_sorted = plot_data[sort_order]
cct_sorted = cct[sort_order]
bus_sorted = [bus_idxs[k] for k in sort_order]
type_sorted = [bus_type[bus_idxs[k]] for k in sort_order]

fig, ax = plt.subplots(figsize=(10, 5.5))

im = ax.imshow(
    plot_sorted, aspect="auto", cmap="RdYlGn_r",
    extent=[tc_values[0] * 1000, tc_values[-1] * 1000,
            n_bus - 0.5, -0.5],
    interpolation="bicubic", vmin=0, vmax=180,
)

# Overlay 90-degree contour
tc_grid, bus_grid = np.meshgrid(tc_values * 1000, np.arange(n_bus))
cs = ax.contour(tc_grid, bus_grid, plot_sorted,
                levels=[90], colors=["white"],
                linewidths=2.0, linestyles="--")
ax.clabel(cs, fmt="90°", fontsize=9, colors="white")

# Mark CCT boundary with white dots
for idx_i in range(n_bus):
    ax.plot(cct_sorted[idx_i] * 1000, idx_i, "o",
            color="white", markersize=6, markeredgecolor="k",
            markeredgewidth=0.8, zorder=5)

# Color-coded bus type labels
type_marker = {"gen": "G", "tie": "T", "load": "L"}
type_color = {"gen": "red", "tie": "blue", "load": "green"}
ylabels = [f"Bus {b} ({type_marker[type_sorted[k]]})"
           for k, b in enumerate(bus_sorted)]

ax.set_yticks(range(n_bus))
ax.set_yticklabels(ylabels, fontsize=9)
for tick_label, t in zip(ax.get_yticklabels(), type_sorted):
    tick_label.set_color(type_color[t])
    tick_label.set_fontweight("bold")

ax.set_xlabel("Fault clearing time [ms]", fontsize=11)
ax.set_title("Maximum rotor angle separation, sorted by CCT", fontsize=13)

cb = plt.colorbar(im, ax=ax, shrink=0.85, pad=0.02)
```

(continues on next page)

(continued from previous page)

```
cb.set_label("Max angle separation [deg]", fontsize=10)

legend_handles = [
    mpatches.Patch(color=type_color["gen"], label="Generator bus (G)"),
    mpatches.Patch(color=type_color["tie"], label="Tie-line bus (T)"),
    mpatches.Patch(color=type_color["load"], label="Load bus (L)"),
    plt.Line2D([0], [0], marker="o", color="w", markerfacecolor="white",
               markeredgecolor="k", markersize=7, label="CCT boundary"),
]
ax.legend(handles=legend_handles, loc="lower right", fontsize=8,
          framealpha=0.9, edgecolor="gray")

plt.tight_layout()
```



Spatial Distribution of CCT

The network topology diagram below maps the CCT values onto the physical layout of the Kundur two-area system. Each bus is colored according to its CCT, with the value annotated below the node. Generator buses are represented by squares, and non-generator buses by circles.

```
# Kundur two-area layout
bus_pos = {
    1: (0.5, 2.0),   2: (0.5, 0.5),
    3: (6.5, 2.0),   4: (6.5, 0.5),
    5: (1.8, 1.25),  6: (2.8, 1.25),
    7: (3.5, 1.65),  8: (3.5, 0.85),
```

(continues on next page)

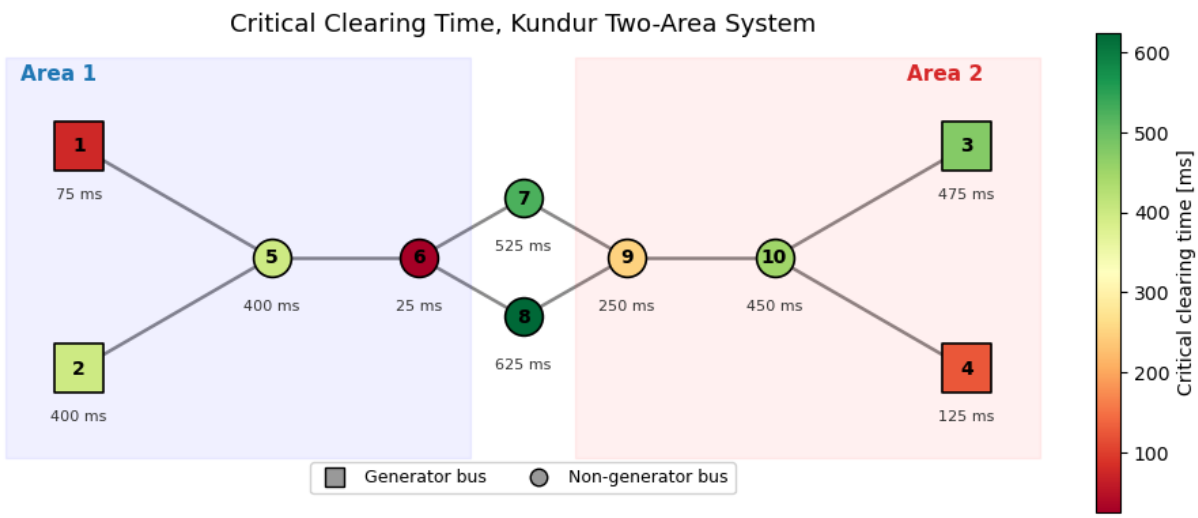
(continued from previous page)

```
    9: (4.2, 1.25),  10: (5.2, 1.25),
}

branches = [
    (1, 5), (2, 5), (5, 6), (6, 7), (6, 8),
    (7, 9), (8, 9), (9, 10), (10, 3), (10, 4),
]

cct_dict = {bus_idxs[i]: cct[i] for i in range(n_bus)}
cct_vals = np.array([cct_dict[b] for b in bus_idxs])

fig, ax = plt.subplots(figsize=(10, 5))

for b1, b2 in branches:
    x = [bus_pos[b1][0], bus_pos[b2][0]]
    y = [bus_pos[b1][1], bus_pos[b2][1]]
    ax.plot(x, y, "-", color="#888888", linewidth=1.8, zorder=1)

cmap = plt.cm.RdYlGn
norm = Normalize(vmin=cct_vals.min() * 1000, vmax=cct_vals.max() * 1000)

for b in bus_idxs:
    x, y = bus_pos[b]
    c = cmap(norm(cct_dict[b] * 1000))
    size = 700 if b in gen_buses else 400
    marker = "s" if b in gen_buses else "o"
    ax.scatter(x, y, s=size, c=[c], marker=marker,
               edgecolors="k", linewidths=1.2, zorder=3)
    ax.annotate(f"{b}", (x, y), fontsize=10, fontweight="bold",
                ha="center", va="center", zorder=4)
    ax.annotate(f"{cct_dict[b]*1000:.0f} ms", (x, y - 0.28),
                fontsize=8, ha="center", va="top", color="#333333", zorder=4)

# Area shading
ax.fill([0.0, 3.15, 3.15, 0.0], [-0.1, -0.1, 2.6, 2.6],
        alpha=0.06, color="blue")
ax.fill([3.85, 7.0, 7.0, 3.85], [-0.1, -0.1, 2.6, 2.6],
        alpha=0.06, color="red")
ax.text(0.1, 2.45, "Area 1", fontsize=11, color="#1f77b4", fontweight="bold")
ax.text(6.1, 2.45, "Area 2", fontsize=11, color="#d62728", fontweight="bold")

sm = ScalarMappable(cmap=cmap, norm=norm)
cb = plt.colorbar(sm, ax=ax, shrink=0.75, pad=0.02)
cb.set_label("Critical clearing time [ms]", fontsize=10)

legend_handles = [
    plt.Line2D([0], [0], marker="s", color="w", markerfacecolor="#999",
               markeredgecolor="k", markersize=10, label="Generator bus"),
    plt.Line2D([0], [0], marker="o", color="w", markerfacecolor="#999",
               markeredgecolor="k", markersize=9, label="Non-generator bus"),
]
ax.legend(handles=legend_handles, loc="lower center", fontsize=9,
```

(continues on next page)

(continued from previous page)

```
          framealpha=0.9, ncol=2)

ax.set_xlim(-0.2, 7.2)
ax.set_ylim(-0.4, 2.7)
ax.set_aspect("equal")
ax.axis("off")
ax.set_title("Critical Clearing Time, Kundur Two-Area System", fontsize=13)

plt.tight_layout()
```



The results reveal a consistent ordering: tie-line midpoint buses (7, 8) have the longest CCTs, followed by interior generator and load buses (2, 3, 5, 10), then boundary generator buses (1, 4), and finally the Area 1 border bus (6) with the shortest CCT at 25 ms. A notable asymmetry exists between the two area border buses: bus 6 and bus 9 occupy topologically symmetric positions, yet their CCTs differ by an order of magnitude. The source of this asymmetry is not immediately apparent from the topology alone and likely involves the pre-fault tie-line power flow direction, specific line impedances, and load distribution.

Rotor Angle Trajectories at the Stability Boundary

To illustrate the transition from stable to unstable behavior, rotor angle trajectories are plotted for a fault at bus 8 at four clearing times that straddle the CCT. The angles are expressed relative to the center of inertia (COI), which removes the common-mode swing and isolates the inter-machine separation.

For the final (unstable) case, the stability criterion is disabled so that the simulation continues beyond the point of instability, thereby capturing the full divergent trajectory.

```
demo_bus = 8
demo_tcs = [0.400, 0.525, 0.600, 0.650]
demo_results = {}
M_vals = np.array(ss_ref.GENROU.M.v)

for tc in demo_tcs:
```

(continues on next page)

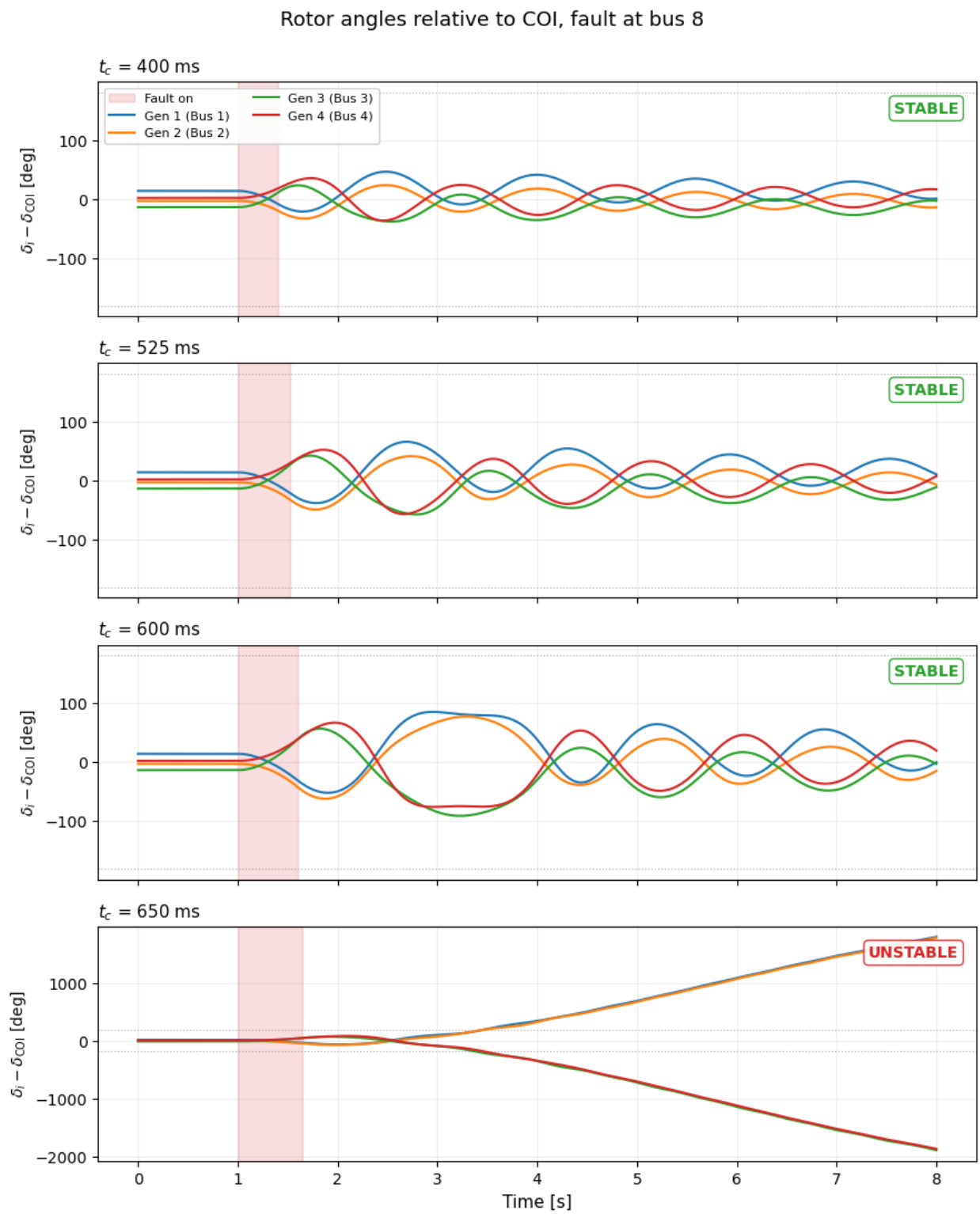
(continued from previous page)

```
    ss = andes.load(case_path, setup=False, no_output=True)
    ss.Toggle.u.v[0] = 0
    ss.add("Fault", bus=demo_bus, tf=1.0, tc=1.0 + tc, xf=1e-4)
    ss.setup()
    ss.config.warn_abnormal = 0
    ss.PFlow.run()
    ss.TDS.config.tf = 8.0
    ss.TDS.config.no_tqdm = 1
    # Disable criteria for the last case to capture full divergent trajectory
    if tc == demo_tcs[-1]:
        ss.TDS.config.criteria = 0
    ss.TDS.run()
    delta_df = ss.TDS.get_timeseries(ss.GENROU.delta)
    demo_results[tc] = np.rad2deg(delta_df)

# Compute COI-relative angles
coi_results = {}
for tc, df in demo_results.items():
    delta_arr = df.values
    coi = np.average(delta_arr, axis=1, weights=M_vals)
    relative = delta_arr - coi[:, None]
    coi_results[tc] = pd.DataFrame(relative, index=df.index, columns=df.
↪columns)

gen_labels = [f"Gen {idx} (Bus {b})" for idx, b in zip(gen_idxs, gen_buses)]
colors = ["#1f77b4", "#ff7f0e", "#2ca02c", "#d62728"]

fig, axes = plt.subplots(4, 1, figsize=(9, 11), sharex=True, sharey=False)

# Share y-axis among the first 3 stable panels; let the last panel float
for ax in axes[1:3]:
    ax.sharey(axes[0])

for ax, tc in zip(axes, demo_tcs):
    df = coi_results[tc]

    ax.axvspan(1.0, 1.0 + tc, alpha=0.15, color="#d62728", zorder=0,
               label="Fault on" if tc == demo_tcs[0] else None)

    ax.axhline(180, color="gray", linewidth=0.8, linestyle=":", alpha=0.6)
    ax.axhline(-180, color="gray", linewidth=0.8, linestyle=":", alpha=0.6)

    for k, col in enumerate(df.columns):
        ax.plot(df.index, df[col], linewidth=1.5, color=colors[k],
                label=gen_labels[k] if tc == demo_tcs[0] else None)

    max_sep = df.max().max() - df.min().min()
    if max_sep < 180:
        status_text, status_color = "STABLE", "#2ca02c"
    else:
        status_text, status_color = "UNSTABLE", "#d62728"
```

(continues on next page)

(continued from previous page)

```
    ax.text(0.98, 0.92, status_text, transform=ax.transAxes,
            fontsize=10, fontweight="bold", color=status_color,
            ha="right", va="top",
            bbox=dict(boxstyle="round,pad=0.3", facecolor="white",
                      edgecolor=status_color, alpha=0.9))

    ax.set_title(f"$t_c$ = {tc*1000:.0f} ms", fontsize=11, loc="left")
    ax.set_ylabel(r"$\delta_i - \delta_{\mathrm{COI}}$ [deg]", fontsize=10)
    ax.grid(True, alpha=0.2)

axes[-1].set_xlabel("Time [s]", fontsize=11)
axes[0].legend(loc="upper left", fontsize=8, ncol=2, framealpha=0.9)

fig.suptitle(f"Rotor angles relative to COI, fault at bus {demo_bus}",
             fontsize=13, y=0.995)
plt.tight_layout()
```



The four panels confirm the expected transition from stable to unstable behavior. For clearing times well below the CCT, the rotor angles undergo damped oscillations relative to the COI and converge to a new equilibrium. As the clearing time approaches the CCT, the oscillation amplitude increases and the return to equilibrium becomes slower. Beyond the CCT, the generators separate irreversibly, indicating loss of synchronism.

Extensions

Several directions can be pursued to extend this example:

- **Fault severity**: the fault reactance x_f can be increased to model less severe faults, and the resulting shift of the CCT boundary can be observed.
- **Line trip protection**: the built-in `Toggle` event can be re-enabled to simulate a fault cleared by tripping a line, which is the standard protection scheme in practice.
- **Load level sweep**: all loads can be scaled by a common factor, and the CCT map can be recomputed to study how loading affects transient stability margins.
- **Machine learning classifier**: the (bus, clearing time, outcome) data can be used as labeled training samples for a classifier that predicts stability from pre-fault features alone.

3.1.2 Low-Inertia Frequency Stability

As inverter-based resources displace synchronous generators, the total rotational inertia of a power system declines. Since inverter-based resources do not inherently contribute inertial response, the system frequency becomes increasingly sensitive to generation-load imbalances. Two metrics characterize this degradation: the rate of change of frequency (RoCoF) immediately following a disturbance, and the frequency nadir, the minimum frequency reached before governor action arrests the decline.

This notebook quantifies the relationship between system inertia and frequency stability on the IEEE 14-bus system (5 GENROU generators with AVR and governor controls) by:

1. Scaling the inertia constants of all generators by a factor α to emulate progressive renewable penetration.
2. Tripping the largest generator at each inertia level and recording the rotor speed transient.
3. Computing the center-of-inertia (COI) frequency and comparing RoCoF and nadir across scenarios.

```
%matplotlib inline

import numpy as np
import pandas as pd
from matplotlib import pyplot as plt

import andes

andes.config_logger(stream_level=30)
```

System Setup

The IEEE 14-bus system contains five GENROU synchronous generators equipped with AVR exciters and turbine governors. The presence of these controllers produces well-damped transient responses that clearly illustrate the effect of inertia on frequency dynamics.

```
case_path = andes.get_case("ieee14/ieee14_full.xlsx")
ss_ref = andes.load(case_path)
ss_ref.PFlow.run()

gen_idxs = list(ss_ref.GENROU.idx.v)
gen_buses = list(ss_ref.GENROU.bus.v)
M_base = ss_ref.GENROU.M.v.copy()

gen_info = pd.DataFrame({"gen_idx": gen_idxs, "bus": gen_buses, "M": M_base})
gen_info.sort_values("M", ascending=False)
```

```
    gen_idx  bus     M
1  GENROU_2    2  13.0
3  GENROU_4    6  10.0
2  GENROU_3    3  10.0
4  GENROU_5    8  10.0
0  GENROU_1    1   8.0
```

Frequency Response Under Varying Inertia

The largest generator by inertia is tripped at $t = 1$ s using a `Toggle` device. Seven inertia scaling factors from $\alpha = 1.0$ down to $\alpha = 0.15$ are simulated. The inertia constant M of every generator is scaled by α before each time-domain run.

```
trip_idx = gen_idxs[int(np.argmax(M_base))]

inertia_scales = [1.0, 0.7, 0.5, 0.4, 0.3, 0.2, 0.15]
results = {}

for alpha in inertia_scales:
    ss = andes.load(case_path, setup=False)
    ss.add("Toggle", model="GENROU", dev=trip_idx, t=1.0)
    ss.setup()
    ss.config.warn_abnormal = 0
    ss.GENROU.M.v[:] = M_base * alpha

    ss.PFlow.run()
    ss.TDS.config.tf = 8
    ss.TDS.config.no_tqdm = 1
    ss.TDS.run()

    results[alpha] = ss.TDS.get_timeseries(ss.GENROU.omega)
```

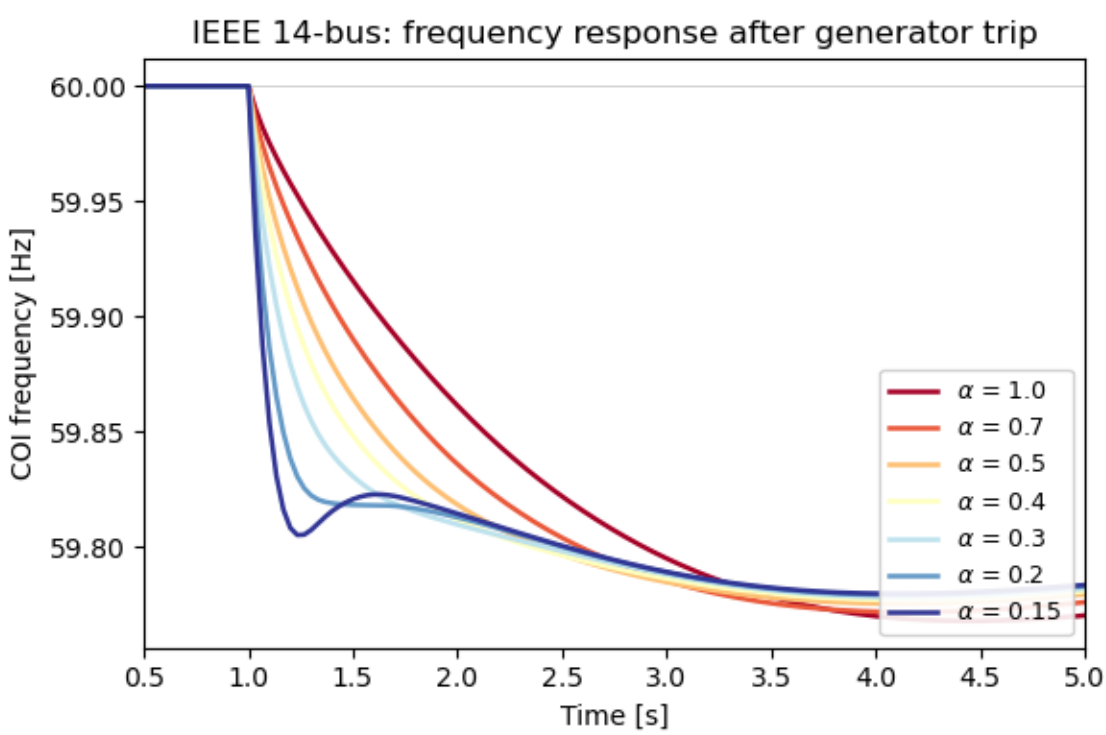
COI Frequency After Generator Trip

The center-of-inertia (COI) frequency is computed as the inertia-weighted average of all generator rotor speeds:

$$f_{\mathrm{COI}} = \left(\sum_i \frac{M_i}{\sum_j M_j} \omega_i \right) \times 60 \text{ Hz}$$

Since the swing equation relates $M_i \, d\omega_i / dt$ to the power imbalance at each machine, the COI frequency directly reflects the aggregate generation-load mismatch following the disturbance.

```
weights = M_base / M_base.sum()
cmap = plt.cm.RdYlBu

fig, ax = plt.subplots(figsize=(6, 4))

for i, (alpha, omega_df) in enumerate(results.items()):
    f_coi = (omega_df.values * weights).sum(axis=1) * 60
    color = cmap(i / (len(results) - 1))
    ax.plot(omega_df.index, f_coi, color=color, linewidth=1.8,
            label=f"$\\alpha$ = {alpha}")

ax.set_xlabel("Time [s]")
ax.set_ylabel("COI frequency [Hz]")
ax.set_title("IEEE 14-bus: frequency response after generator trip")
ax.legend(loc="lower right", fontsize=9, framealpha=0.9)
ax.set_xlim(0.5, 5.0)
ax.axhline(60.0, color="black", linewidth=0.4, alpha=0.3)

plt.tight_layout()
```



As the inertia scaling factor decreases from blue ($\alpha = 1.0$) to red ($\alpha = 0.15$), the initial rate of frequency decline steepens progressively. The curves fan out clearly in the first two seconds following the disturbance, and the well-damped oscillations produced by the AVR and governor controls confirm that the transient is dominated by the inertial response rather than by sustained electromechanical modes.

Quantitative Metrics

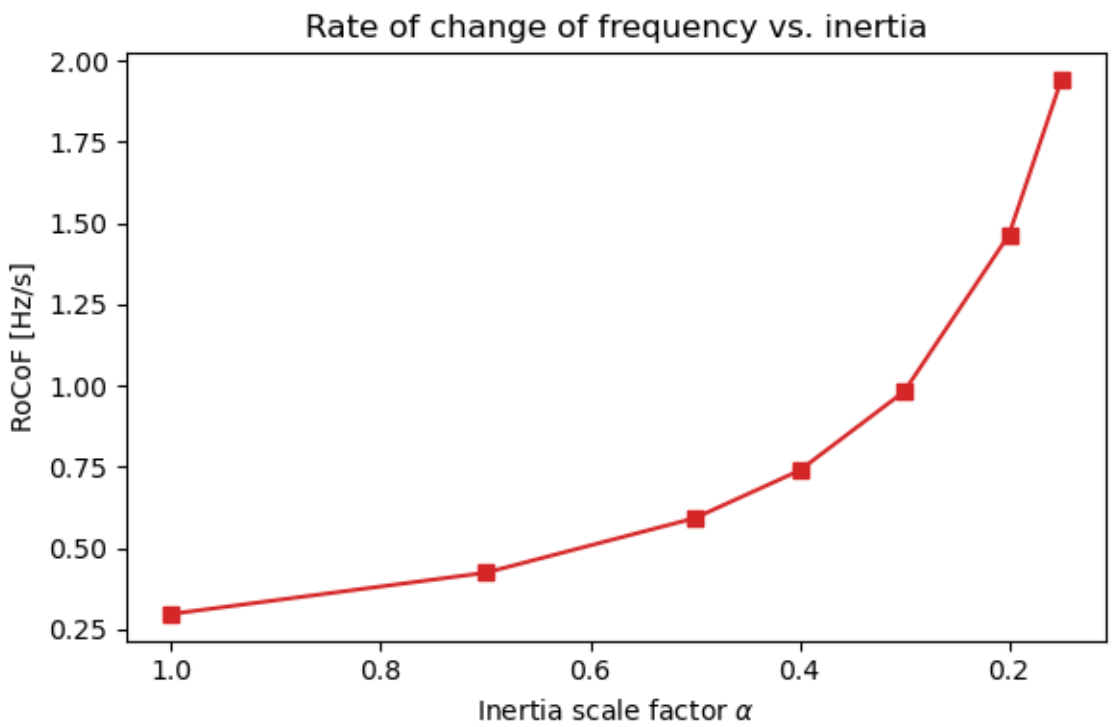
The RoCoF is estimated as the maximum absolute rate of frequency change within the first second after the disturbance. The nadir is the minimum COI frequency reached before governor action arrests the decline.

```
metrics = []
for alpha, omega_df in results.items():
    f_coi = (omega_df.values * weights).sum(axis=1) * 60
    t = omega_df.index.values

    nadir = f_coi[t > 1.0].min()

    post = (t >= 1.0) & (t <= 2.0)
    t_post, f_post = t[post], f_coi[post]
    rocof = np.max(np.abs(np.diff(f_post) / np.diff(t_post)))

    metrics.append({"α": alpha, "Nadir [Hz]": nadir, "RoCoF [Hz/s]": rocof})

metrics_df = pd.DataFrame(metrics).set_index("α")
metrics_df
```

```
      Nadir [Hz]  RoCoF [Hz/s]
α
1.00   59.767679      0.297080
0.70   59.771490      0.423739
0.50   59.775179      0.592006
0.40   59.776735      0.738667
0.30   59.777930      0.981922
0.20   59.778977      1.464048
0.15   59.779485      1.940406
```

RoCoF vs. Inertia Level

```
fig, ax = plt.subplots(figsize=(6, 4))

ax.plot(metrics_df.index, metrics_df["RoCoF [Hz/s]"], "s-", color="tab:red",
        markersize=6)
ax.set_xlabel("Inertia scale factor $\\alpha$")
ax.set_ylabel("RoCoF [Hz/s]")
ax.set_title("Rate of change of frequency vs. inertia")
ax.invert_xaxis()

plt.tight_layout()
```



Reducing the system inertia monotonically increases the RoCoF following the generator trip. From $\alpha = 1.0$ to $\alpha = 0.15$, the RoCoF increases by approximately six times. Although the IEEE 14-bus system is small, the same qualitative behavior applies to larger interconnections: as synchronous generation is displaced by inverter-based resources without synthetic inertia, the system frequency becomes increasingly vulnerable to sudden imbalances.

Extend This Example

- **Add synthetic inertia**: install IBR models with frequency-watt droop (via REECA1) and compare the RoCoF improvement at low α.
- **Vary the contingency size**: trip different generators to study how disturbance magnitude interacts with inertia levels.
- **Sweep inertia continuously**: use a finer grid of α values to trace the RoCoF curve more precisely.
- **Add UFLS relays**: implement under-frequency load shedding and observe the threshold at which it is triggered as inertia decreases.

3.1.3 Forced Oscillation Source Localization

Forced oscillations in power systems arise when a periodic disturbance, such as a malfunctioning governor valve, injects energy at a fixed frequency. Unlike natural oscillations, forced oscillations persist as long as the source remains active, and resonance can amplify the response at generators far from the source. Locating the source from measurements alone is therefore a non-trivial problem.

This notebook demonstrates two PMU-based methods on a 179-bus WECC system with 29 generators:

1. **Energy flow analysis**: the dissipating energy flow (DEF) method computes $E_i = \int \Delta P_{e,i} \cdot \Delta\omega_i \, dt$ at each generator; the source is the only generator with a positive value [1, 2].
2. **Phase difference analysis**: the P_e–ω phase difference at the source is a clear outlier relative to all other generators [2].

Both methods use only bus frequency and electrical power, quantities directly observable from phasor measurement units.

References

[1] S. Maslennikov, B. Wang, and E. Litvinov, "Dissipating energy flow method for locating the source of sustained oscillations," *Int. J. Elect. Power Energy Syst.*, vol. 88, pp. 55–62, Jun. 2017.

[2] L. Chen, Y. Min, and W. Hu, "An energy-based method for location of power system oscillation source," *IEEE Trans. Power Syst.*, vol. 28, no. 2, pp. 828–836, May 2013.

```
%matplotlib inline

import numpy as np
from scipy import signal
from pathlib import Path
from matplotlib import pyplot as plt

import andes

andes.config_logger(stream_level=30)
```

System Setup

The WECC 179-bus system is loaded from PSS/E format files originally provided by Kai Sun's Oscillation Test Cases Library. The model contains 29 GENCLS generators with uniform damping ($D = 4$). A TGOV1 governor is added to each generator to provide realistic mechanical power dynamics, and BusFreq devices are installed at all generator buses.

```
case_dir = Path("cases/wecc179")
ss = andes.load(str(case_dir / "wecc179.raw"),
                addfile=str(case_dir / "wecc179.dyr"), setup=False)

gen_buses = list(ss.GENCLS.bus.v)
gen_idxs = list(ss.GENCLS.idx.v)
n_gen = len(gen_idxs)
```

(continues on next page)

(continued from previous page)

```
for idx in gen_idxs:
    ss.add("TGOV1", syn=idx, R=0.05, T1=0.5, T2=1.0,
           T3=2.0, Dt=0.0, VMAX=99, VMIN=0)

for bus in gen_buses:
    ss.add("BusFreq", bus=bus)

ss.setup()
ss.PFlow.run()
```

```
True
```

Periodic Disturbance Injection

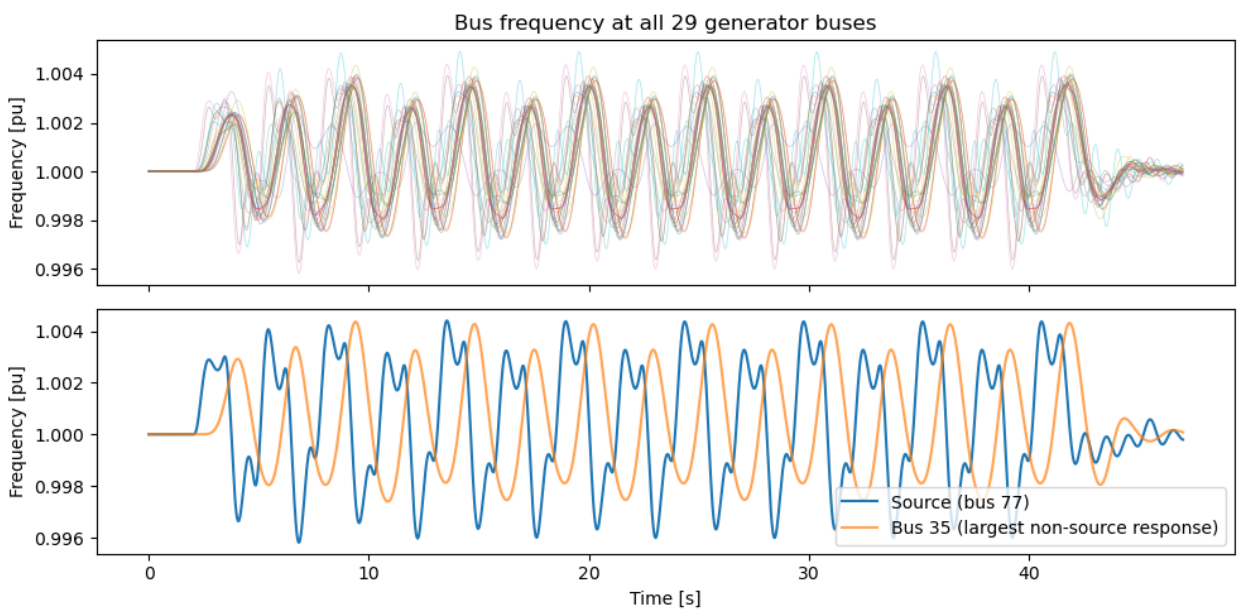
A square-wave disturbance at 0.37 Hz is applied to the auxiliary power input of the generator at bus 77. This frequency coincides with a natural electromechanical mode of the system, producing resonance. Due to resonance, generators far from the source may oscillate with comparable or even larger amplitude, making amplitude-based source identification unreliable.

```
source_bus = 77
source_idx = dict(zip(gen_buses, gen_idxs))[source_bus]

f_osc = 0.37
half_period = 1.0 / (2 * f_osc)
amplitude = 0.03
t_start, t_end = 2.0, 42.0

ss.config.warn_abnormal = 0
ss.TDS.config.tf = t_start
ss.TDS.config.no_tqdm = 1
ss.TDS.run()

t = t_start
positive = True
while t + half_period <= t_end:
    ss.SynGen.set_paux(ss, source_idx, amplitude if positive else -amplitude)
    t += half_period
    ss.TDS.config.tf = t
    ss.TDS.run()
    positive = not positive

ss.SynGen.set_paux(ss, source_idx, 0.0)
ss.TDS.config.tf = t_end + 5.0
ss.TDS.run()
```

```
True
```

Observing the Forced Oscillation

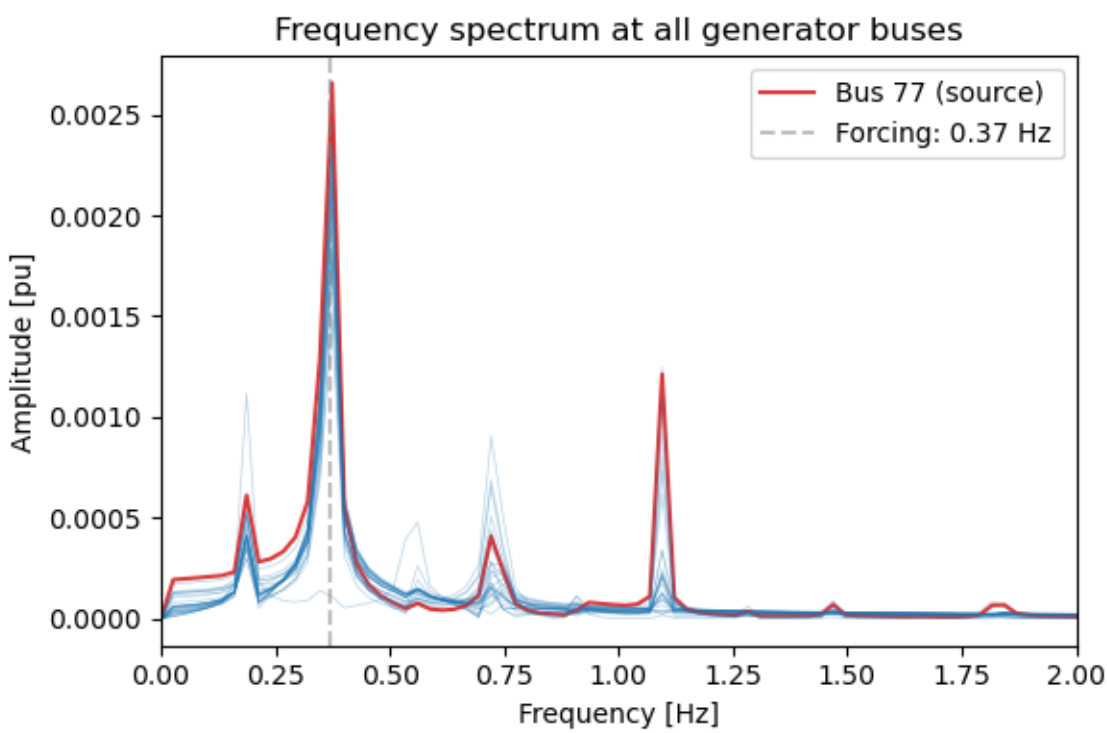
```
ts_bf = ss.TDS.get_timeseries(ss.BusFreq.f)
bf_buses = list(ss.BusFreq.bus.v)

fig, axes = plt.subplots(2, 1, figsize=(10, 5), sharex=True)

for i in range(len(bf_buses)):
    axes[0].plot(ts_bf.index, ts_bf.iloc[:, i], alpha=0.4, linewidth=0.5)
axes[0].set_ylabel("Frequency [pu]")
axes[0].set_title("Bus frequency at all 29 generator buses")

source_col = bf_buses.index(source_bus)
remote_col = bf_buses.index(35)

axes[1].plot(ts_bf.index, ts_bf.iloc[:, source_col],
             label=f"Source (bus {source_bus})")
axes[1].plot(ts_bf.index, ts_bf.iloc[:, remote_col],
             label="Bus 35 (largest non-source response)", alpha=0.7)
axes[1].set_ylabel("Frequency [pu]")
axes[1].set_xlabel("Time [s]")
axes[1].legend()

plt.tight_layout()
```



All 29 generator buses exhibit oscillations at the forcing frequency. The amplitude at bus 35 exceeds that at the source (bus 77), illustrating the resonance amplification effect. This behavior renders simple amplitude comparison unreliable for source identification and motivates the energy-based and phase-based methods developed below.

Spectral Analysis

```
time = ts_bf.index.values
mask = (time >= 5.0) & (time <= t_end)
t_ss = time[mask]
dt = np.median(np.diff(t_ss))
N = len(t_ss)

bf_ss = ts_bf.loc[mask].values
freqs = np.fft.rfftfreq(N, d=dt)

fig, ax = plt.subplots(figsize=(6, 4))
for i, bus in enumerate(bf_buses):
    spectrum = np.abs(np.fft.rfft(bf_ss[:, i] - bf_ss[:, i].mean())) * 2 / N
    is_src = bus == source_bus
    ax.plot(freqs, spectrum,
            alpha=0.9 if is_src else 0.3,
            linewidth=1.5 if is_src else 0.5,
            color="#d62728" if is_src else "#1f77b4",
            label=f"Bus {bus} (source)" if is_src else None)

ax.set_xlim(0, 2.0)
ax.set_xlabel("Frequency [Hz]")
ax.set_ylabel("Amplitude [pu]")
ax.set_title("Frequency spectrum at all generator buses")
ax.axvline(f_osc, color="gray", linestyle="--", alpha=0.5,
           label=f"Forcing: {f_osc} Hz")
ax.legend()
plt.tight_layout()
```



A sharp spectral peak at 0.37 Hz is present at all generator buses, confirming the forced oscillation frequency. The peak amplitudes are comparable across buses, reinforcing the observation that amplitude alone cannot identify the source.

Energy-Based Source Localization

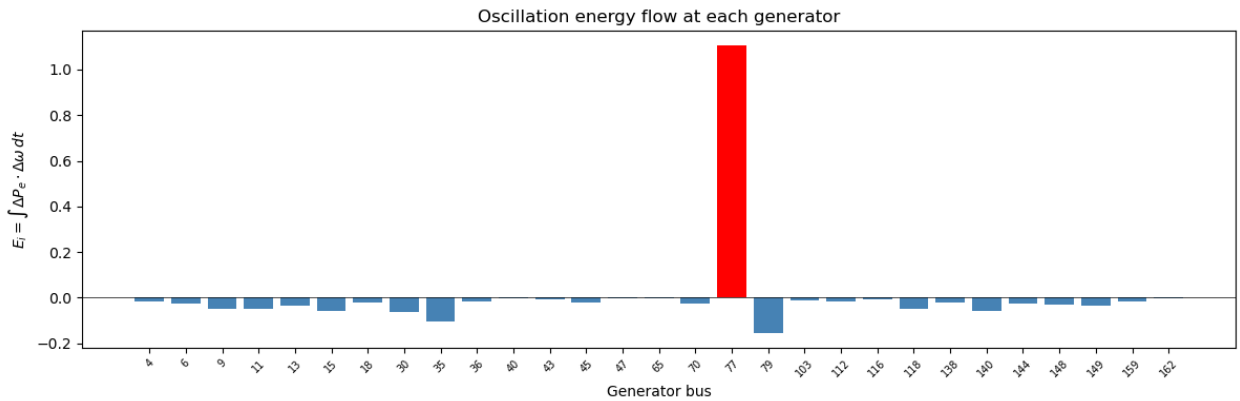
The oscillation energy flow at each generator is computed as

$$E_i = \int \Delta P_{e,i}(t) \cdot \Delta\omega_i(t)\, dt$$

where $\Delta P_{e,i}$ and $\Delta\omega_i$ are bandpass-filtered oscillatory components of electrical power and rotor speed. A generator that injects oscillation energy into the network (the source) yields a positive value, while generators that absorb energy yield negative values.

```
ts_Pe = ss.TDS.get_timeseries(ss.GENCLS.Pe)
ts_omega = ss.TDS.get_timeseries(ss.GENCLS.omega)

Pe_ss = ts_Pe.loc[mask].values
omega_ss = ts_omega.loc[mask].values

fs = 1.0 / dt
sos = signal.butter(4, [max(f_osc - 0.1, 0.05), f_osc + 0.1],
                    btype="bandpass", fs=fs, output="sos")

energy = np.zeros(n_gen)
for j in range(n_gen):
    pe_filt = signal.sosfiltfilt(sos, Pe_ss[:, j])
    om_filt = signal.sosfiltfilt(sos, omega_ss[:, j] - 1.0)
    energy[j] = np.trapezoid(pe_filt * om_filt, t_ss)
```

```
bus_labels = [str(b) for b in gen_buses]
colors = ["red" if b == source_bus else "steelblue" for b in gen_buses]

fig, ax = plt.subplots(figsize=(12, 4))
ax.bar(bus_labels, energy, color=colors)
ax.set_xlabel("Generator bus")
ax.set_ylabel(r"$E_i = \int \Delta P_e \cdot \Delta\omega\, dt$")
ax.set_title("Oscillation energy flow at each generator")
ax.axhline(0, color="black", linewidth=0.5)
plt.xticks(rotation=45, fontsize=7)
plt.tight_layout()
```



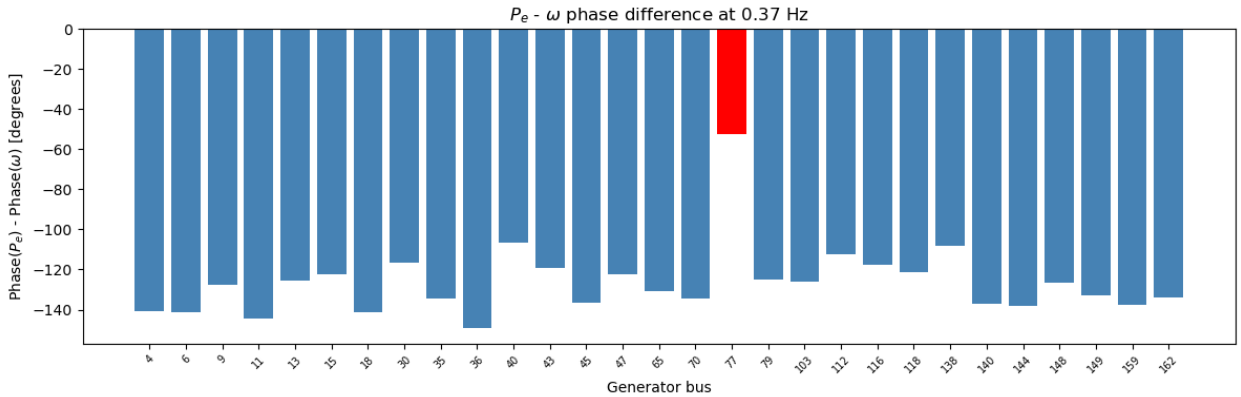
The source generator at bus 77 (red) is the only generator with a positive energy flow, indicating that it injects oscillation energy into the network. All remaining 28 generators have negative values, meaning they absorb energy from the oscillation. The margin is decisive: the source energy exceeds +1.1, while the next-closest generator is near zero.

Phase-Based Confirmation

The phase difference between electrical power and rotor speed oscillations at the forcing frequency provides an independent confirmation of the source location. At the source, the P_e–ω phase difference deviates substantially from the cluster formed by all other generators.

```
f_idx = np.argmin(np.abs(freqs - f_osc))

phase_diff = np.zeros(n_gen)
for j in range(n_gen):
    fft_pe = np.fft.rfft(Pe_ss[:, j] - Pe_ss[:, j].mean())
    fft_om = np.fft.rfft(omega_ss[:, j] - omega_ss[:, j].mean())
    diff = np.angle(fft_pe[f_idx]) - np.angle(fft_om[f_idx])
    phase_diff[j] = np.degrees(np.arctan2(np.sin(diff), np.cos(diff)))
```

```
fig, ax = plt.subplots(figsize=(12, 4))
ax.bar(bus_labels, phase_diff, color=colors)
ax.set_xlabel("Generator bus")
ax.set_ylabel("Phase($P_e$) - Phase($\\omega$) [degrees]")
ax.set_title(f"$P_e$ - $\\omega$ phase difference at {f_osc} Hz")
plt.xticks(rotation=45, fontsize=7)
plt.tight_layout()
```



The source generator at bus 77 exhibits a P_e–ω phase difference of approximately $-52°$, while all other generators cluster between $-107°$ and $-149°$. This distinct outlier position provides unambiguous identification of the source, independent of and consistent with the energy flow result.

Extend This Example

- **Change the forcing frequency** away from resonance and observe how the energy flow margins change.
- **Relocate the source** to a different generator and verify that both methods correctly identify the new location.
- **Add measurement noise** to test robustness under realistic PMU conditions.
- **Apply to field PMU data** from actual forced oscillation events documented in Kai Sun's Oscillation Test Cases Library.

3.1.4 Physics-Informed Surrogate from Symbolic Equations

Surrogate models that respect the governing physics can accelerate repeated simulation studies such as contingency screening and sensitivity analysis. A key challenge is encoding the differential- algebraic equations (DAEs) of the power system into a differentiable form suitable for neural network training.

ANDES stores every model equation as a symbolic string. Calling `model.prepare()` parses these strings into SymPy expressions, which can then be manipulated algebraically and converted to PyTorch functions via `sp.lambdify`. This notebook demonstrates the complete pipeline:

1. **Extract** the GENCLS differential and algebraic equations from ANDES.
2. **Eliminate** the 12 internal algebraic variables with `sp.solve`.
3. **Convert** the reduced ODE residual to a differentiable PyTorch function with `sp.lambdify`.
4. **Train** a parameterized Physics-Informed Neural Network (PINN) using the auto-derived residual as the physics loss.

No physics is hand-coded in PyTorch. The entire residual is derived programmatically from the ANDES model definition.

```
%matplotlib inline

import time

import numpy as np
import sympy as sp
import torch
import torch.nn as nn
from matplotlib import pyplot as plt

import andes

andes.config_logger(stream_level=40)

DEVICE = "cuda" if torch.cuda.is_available() else "cpu"
DTYPE = torch.float64

# SymPy-to-PyTorch mapping for sp.lambdify
TORCH_MODULES = {
    "sin": torch.sin, "cos": torch.cos, "exp": torch.exp,
    "log": torch.log, "sqrt": torch.sqrt, "Abs": torch.abs,
    "sign": torch.sign, "asin": torch.asin, "acos": torch.acos,
    "atan": torch.atan,
    "pi": torch.tensor(np.pi, dtype=torch.float64),
    "re": lambda x: x,
    "im": lambda x: torch.zeros_like(x),
}

print(f"Device: {DEVICE}")
```

```
Device: cuda
```

Step 1 — Extract Symbolic Equations

A single-machine infinite-bus (SMIB) system is constructed using the ANDES v2 API. After power flow and TDS initialization, calling `gen.prepare()` parses the `e_str` attributes of every GENCLS variable into SymPy expressions stored in `syms.f_list` (differential) and `syms.g_list` (algebraic).

```
ss = andes.System(default_config=True)
ss.undill()

ss.add("Bus", idx=1, name="Gen Bus", Vn=20.0, v0=1.0)
ss.add("Bus", idx=2, name="Inf Bus", Vn=20.0, v0=1.0)
ss.add("Line", idx=1, bus1=1, bus2=2, Vn1=20.0, Vn2=20.0,
       r=0.0, x=0.3, b=0.0)
ss.add("PV", idx=1, bus=1, Vn=20.0, p0=0.5, q0=0.0, v0=1.0)
ss.add("Slack", idx=2, bus=2, Vn=20.0, v0=1.0)
ss.add("GENCLS", idx=1, bus=1, gen=1, Sn=100, Vn=20.0,
       M=6.0, D=20.0, xd1=0.3, ra=0.0)
```

(continues on next page)

(continued from previous page)

```
ss.setup()
ss.PFlow.run()
ss.TDS.init()

gen = ss.GENCLS
gen.prepare()

state_names = list(gen.cache.states_and_ext.keys())
algeb_names = list(gen.cache.algebs_and_ext.keys())

print("Differential equations (f_list)")
for i, name in enumerate(state_names):
    var = gen.cache.states_and_ext[name]
    tc = f" / {var.t_const.name}" if getattr(var, "t_const", None) else ""
    print(f"  d({name})/dt{tc} = {gen.syms.f_list[i]}")

print(f"\nAlgebraic equations ({len(algeb_names)} total, first 4 shown)")
for i in range(min(4, len(algeb_names))):
    print(f"  0 = {gen.syms.g_list[i]}")
```

```
Differential equations (f_list)
  d(delta)/dt = 2*pi*fn*ue*(omega - 1)
  d(omega)/dt / M = ue*(-D*(omega - 1) - te + tm)

Algebraic equations (14 total, first 4 shown)
  0 = Id*xq + psid - vf
  0 = Iq*xq + psiq
  0 = -ue*v*sin(a - delta) - vd
  0 = ue*v*cos(a - delta) - vq
```

Step 2 — Eliminate Algebraic Variables

The GENCLS model has 12 internal algebraic variables (I_d, I_q, v_d, v_q, t_e, etc.) that couple the differential equations to the network. These are eliminated automatically with `sp.solve`, leaving an ODE in the state variables (δ, ω) and the external bus quantities (v, a) that serve as parameters.

```
syms_d = gen.syms.inputs_dict

# Collect internal algebraic equations and variables
int_algeb = [n for n, v in gen.cache.algebs_and_ext.items()
             if not hasattr(v, "model")]
g_eqs = [gen.syms.g_list[i] for i, n in enumerate(algeb_names)
         if n in int_algeb]
g_vars = [syms_d[n] for n in int_algeb]

sol = sp.solve(g_eqs, g_vars, dict=True)[0]

# Substitute into both differential equations
f_delta_reduced = sp.simplify(gen.syms.f_list[0].subs(sol))
```

(continues on next page)

(continued from previous page)

```
f_omega_reduced = sp.simplify(gen.syms.f_list[1].subs(sol))

print(f"Eliminated {len(sol)} algebraic variables")
print(f"\n  d(delta)/dt     = {f_delta_reduced}")
print(f"  M * d(omega)/dt = {f_omega_reduced}")
print(f"\nAuto-derived te = {sp.simplify(sol[syms_d['te']])}")
```

```
Eliminated 12 algebraic variables

  d(delta)/dt     = 2*pi*fn*ue*(omega - 1)
  M * d(omega)/dt = ue*(ue**4*vf0*(ra*ue*v*cos(a - delta) - ra*vf0 +␣
↪v*xq*sin(a - delta)) - (D*(omega - 1) - tm0)*(ra**2*ue**2 + xq**2))/
↪(ra**2*ue**2 + xq**2)

Auto-derived te = ue**4*vf0*(-ra*ue*v*cos(a - delta) + ra*vf0 - v*xq*sin(a -␣
↪delta))/(ra**2*ue**2 + xq**2)
```

Step 3 — Convert to Differentiable PyTorch Functions

The two reduced first-order ODEs from Step 2 are converted directly to PyTorch-compatible functions via `sp.lambdify`. Machine constants (M, D, x'_d) are fixed at their numerical values, while the per-equilibrium coefficients (v_{f0}, v, a, τ_{m0}) are kept symbolic so that the residuals can be evaluated across operating points.

Each `lambdify` call produces a fully differentiable function that supports `torch.autograd`. No manual algebraic manipulation is performed; the ODEs are passed through as ANDES produced them.

```
# Fix machine constants, keep per-equilibrium params symbolic
const_subs = {
    syms_d["M"]: float(gen.M.v[0]),
    syms_d["D"]: float(gen.D.v[0]),
    syms_d["fn"]: float(ss.config.freq),
    syms_d["ue"]: 1.0,
    syms_d["ra"]: 0.0,
    syms_d["xq"]: float(gen.xd1.v[0]),
}

f_delta_num = sp.simplify(f_delta_reduced.subs(const_subs))
f_omega_num = sp.simplify(f_omega_reduced.subs(const_subs))
M_val = float(gen.M.v[0])

print(f"d(delta)/dt = {f_delta_num}")
print(f"d(omega)/dt = ({f_omega_num}) / {M_val}")
```

```
d(delta)/dt = 120.0*pi*(omega - 1)
d(omega)/dt = (-20.0*omega + 1.0*tm0 + 3.33333333333333*v*vf0*sin(a - delta)␣
↪+ 20.0) / 6.0
```

```
delta_s = syms_d["delta"]
omega_s = syms_d["omega"]
```

(continues on next page)

(continued from previous page)

```
vf0_s = syms_d["vf0"]
v_s = syms_d["v"]
a_s = syms_d["a"]
tm0_s = syms_d["tm0"]

args = [delta_s, omega_s, vf0_s, v_s, a_s, tm0_s]

f_delta_torch = sp.lambdify(args, f_delta_num, modules=[TORCH_MODULES, "numpy
↪"])
f_omega_torch = sp.lambdify(args, f_omega_num, modules=[TORCH_MODULES, "numpy
↪"])

# Quick autograd check
d_test = torch.tensor(0.3, dtype=DTYPE, requires_grad=True)
o_test = torch.tensor(1.0, dtype=DTYPE, requires_grad=True)
vf0_t = torch.tensor(float(gen.vf0.v[0]), dtype=DTYPE)
v_t = torch.tensor(float(ss.Bus.v.v[0]), dtype=DTYPE)
a_t = torch.tensor(float(ss.Bus.a.v[0]), dtype=DTYPE)
tm0_t = torch.tensor(float(gen.tm0.v[0]), dtype=DTYPE)

ro = f_omega_torch(d_test, o_test, vf0_t, v_t, a_t, tm0_t)
ro.backward()
print(f"sp.lambdify -> PyTorch OK, autograd: d(f_omega)/d(delta) = {d_test.
↪grad.item():.4f}")
```

```
sp.lambdify -> PyTorch OK, autograd: d(f_omega)/d(delta) = -3.3699
```

Step 4 — Train a Parameterized PINN

The PINN learns the solution surface $(\delta, \omega)(P_m, t)$ across a range of mechanical power setpoints. ANDES TDS generates the ground-truth trajectories (with a 3-cycle fault at $t = 0.1$ s); the auto-derived residuals from Step 3 provide the physics loss.

For each P_m, a separate power flow is solved to obtain the per-equilibrium coefficients (v_{f0}, v, a, τ_{m0}). These are interpolated to each collocation point so that the physics loss reflects the correct equilibrium at every P_m.

Element	Value
Network	3-layer MLP, 256 units, tanh, 2 outputs
Input / Output	$(P_m,\ t) \to (\delta,\ \omega)$
Supervised points	8 000 (20 P_m values $\times$ 400 time steps)
Collocation points	8 000 (random, excluding fault window)
Physics weight	0.01
Optimizer	Adam (5 000 epochs, cosine LR) + L-BFGS refinement

```
def build_smib(Pm, tf=2.0):
    """Build SMIB, apply 3-cycle fault, run TDS."""
```

(continues on next page)

(continued from previous page)

```
    ss_r = andes.System(default_config=True)
    ss_r.undill()
    ss_r.add("Bus", idx=1, name="Gen", Vn=20.0, v0=1.0)
    ss_r.add("Bus", idx=2, name="Inf", Vn=20.0, v0=1.0)
    ss_r.add("Line", idx=1, bus1=1, bus2=2,
             Vn1=20.0, Vn2=20.0, r=0.0, x=0.3, b=0.0)
    ss_r.add("PV", idx=1, bus=1, Vn=20.0, p0=Pm, q0=0.0, v0=1.0)
    ss_r.add("Slack", idx=2, bus=2, Vn=20.0, v0=1.0)
    ss_r.add("GENCLS", idx=1, bus=1, gen=1, Sn=100, Vn=20.0,
             M=6.0, D=20.0, xd1=0.3, ra=0.0)
    ss_r.add("Fault", idx=1, bus=1, tf=0.1, tc=0.15, xf=1e-4)
    ss_r.setup()
    if not ss_r.PFlow.run():
        return None
    ss_r.TDS.config.tf = tf
    ss_r.TDS.run()
    ts_d = ss_r.TDS.get_timeseries(ss_r.GENCLS.delta)
    ts_o = ss_r.TDS.get_timeseries(ss_r.GENCLS.omega)
    if ts_d is None or len(ts_d) < 20:
        return None
    g = ss_r.GENCLS
    return dict(
        t=ts_d.index.values.astype(np.float64),
        delta=ts_d.iloc[:, 0].values.astype(np.float64),
        omega=ts_o.iloc[:, 0].values.astype(np.float64),
        coeff=dict(vf0=float(g.vf0.v[0]), v=float(ss_r.Bus.v.v[0]),
                   a=float(ss_r.Bus.a.v[0]), tm0=float(g.tm0.v[0])),
    )


class SurrogateNet(nn.Module):
    """(Pm, t) -> (delta, omega) with input/output normalization."""

    def __init__(self, widths, lb, ub, y_shift, y_scale):
        super().__init__()
        self.register_buffer("lb", torch.tensor(lb, dtype=DTYPE))
        self.register_buffer("ub", torch.tensor(ub, dtype=DTYPE))
        self.register_buffer("y_shift", torch.tensor(y_shift, dtype=DTYPE))
        self.register_buffer("y_scale", torch.tensor(y_scale, dtype=DTYPE))

        layers = []
        for i in range(len(widths) - 1):
            layers.append(nn.Linear(widths[i], widths[i + 1]))
            if i < len(widths) - 2:
                layers.append(nn.Tanh())
        self.net = nn.Sequential(*layers)

        for m in self.net:
            if isinstance(m, nn.Linear):
                nn.init.xavier_normal_(m.weight)
                nn.init.zeros_(m.bias)
```

(continues on next page)

(continued from previous page)

```
    def forward(self, X):
        H = 2.0 * (X - self.lb) / (self.ub - self.lb + 1e-12) - 1.0
        return self.net(H) * self.y_scale + self.y_shift
```

```
Pm_sweep = np.linspace(0.1, 1.2, 20)
tf = 2.0
n_t = 400

print(f"Generating ground truth: {len(Pm_sweep)} Pm values, "
      f"tf={tf} s, 3-cycle fault at t=0.1 s")
t0 = time.time()

runs, valid_Pm = [], []
coeffs = {"vf0": [], "v": [], "a": [], "tm0": []}

for Pm in Pm_sweep:
    ret = build_smib(Pm, tf=tf)
    if ret is None:
        continue
    runs.append((ret["t"], ret["delta"], ret["omega"]))
    valid_Pm.append(Pm)
    for k in coeffs:
        coeffs[k].append(ret["coeff"][k])

t_vec = np.linspace(0.0, tf, n_t)
usol_delta = np.array([np.interp(t_vec, ti, di) for ti, di, _ in runs])
usol_omega = np.array([np.interp(t_vec, ti, oi) for ti, _, oi in runs])
Pm_vec = np.array(valid_Pm)
for k in coeffs:
    coeffs[k] = np.array(coeffs[k])

print(f"  {usol_delta.shape[0]} operating points x {usol_delta.shape[1]} "
      f"time steps, {time.time()-t0:.1f} s")
```

```
Generating ground truth: 20 Pm values, tf=2.0 s, 3-cycle fault at t=0.1 s
  20 operating points x 400 time steps, 3.3 s
```

```
  0%|          | 0/100 [00:00<?, ?%/s]
```

```
  0%|          | 0/100 [00:00<?, ?%/s]
```

```
  0%|          | 0/100 [00:00<?, ?%/s]
```

```
  0%|          | 0/100 [00:00<?, ?%/s]
```

```
  0%|          | 0/100 [00:00<?, ?%/s]
```

```
  0%|          | 0/100 [00:00<?, ?%/s]
```

```
  0%|          | 0/100 [00:00<?, ?%/s]
```

```
  0%|          | 0/100 [00:00<?, ?%/s]
```

```
  0%|          | 0/100 [00:00<?, ?%/s]
```

```
  0%|          | 0/100 [00:00<?, ?%/s]
```

```
  0%|          | 0/100 [00:00<?, ?%/s]
```

```
  0%|          | 0/100 [00:00<?, ?%/s]
```

```
  0%|          | 0/100 [00:00<?, ?%/s]
```

```
  0%|          | 0/100 [00:00<?, ?%/s]
```

```
  0%|          | 0/100 [00:00<?, ?%/s]
```

```
  0%|          | 0/100 [00:00<?, ?%/s]
```

```
  0%|          | 0/100 [00:00<?, ?%/s]
```

```
  0%|          | 0/100 [00:00<?, ?%/s]
```

```
  0%|          | 0/100 [00:00<?, ?%/s]
```

```
  0%|          | 0/100 [00:00<?, ?%/s]
```

```
np.random.seed(42)
torch.manual_seed(42)

Pm_g, T_g = np.meshgrid(Pm_vec, t_vec, indexing="ij")
X_all = np.column_stack([Pm_g.ravel(), T_g.ravel()])
u_delta = usol_delta.ravel()
u_omega = usol_omega.ravel()

X_u = torch.tensor(X_all, dtype=DTYPE, device=DEVICE)
y_u = torch.tensor(np.column_stack([u_delta, u_omega]),
                   dtype=DTYPE, device=DEVICE)

lb = np.array([Pm_vec[0], t_vec[0]])
ub = np.array([Pm_vec[-1], t_vec[-1]])

# Collocation points (exclude fault window 0.1-0.15 s)
N_coll = 8000
pts = lb + (ub - lb) * np.random.rand(2 * N_coll, 2)
mask = (pts[:, 1] < 0.1) | (pts[:, 1] > 0.15)
```

(continues on next page)

(continued from previous page)

```
pts = pts[mask][:N_coll]
X_f = torch.tensor(pts, dtype=DTYPE, device=DEVICE)

# Per-collocation equilibrium coefficients (interpolated from Pm sweep)
Pm_coll = X_f[:, 0].cpu().numpy()
coll_coeffs = {}
for k in coeffs:
    coll_coeffs[k] = torch.tensor(
        np.interp(Pm_coll, Pm_vec, coeffs[k]),
        dtype=DTYPE, device=DEVICE,
    ).unsqueeze(1)

# Build network
y_mean = np.array([u_delta.mean(), u_omega.mean()])
y_std = np.array([max(u_delta.std(), 1e-6), max(u_omega.std(), 1e-6)])
widths = [2, 256, 256, 256, 2]

model = SurrogateNet(
    widths, lb, ub, y_shift=y_mean, y_scale=y_std,
).to(DEVICE).to(DTYPE)

print(f"Supervised: {X_u.shape[0]} pts,  Collocation: {X_f.shape[0]} pts")
```

```
Supervised: 8000 pts,  Collocation: 8000 pts
```

```
def compute_physics_loss(model, X_f, coll_coeffs, M_val):
    """Physics loss enforcing both first-order ODEs."""
    X = X_f.clone().requires_grad_(True)
    out = model(X)                      # (N, 2): [delta, omega]
    delta_pred = out[:, 0:1]
    omega_pred = out[:, 1:2]

    # d(delta)/dt and d(omega)/dt via autograd
    ones = torch.ones_like(delta_pred)
    g_delta = torch.autograd.grad(delta_pred, X, ones, create_graph=True)[0]
    ddelta_dt = g_delta[:, 1:2]

    g_omega = torch.autograd.grad(omega_pred, X, ones, create_graph=True)[0]
    domega_dt = g_omega[:, 1:2]

    # RHS from auto-derived ANDES equations
    rhs_delta = f_delta_torch(
        delta_pred, omega_pred,
        coll_coeffs["vf0"], coll_coeffs["v"],
        coll_coeffs["a"], coll_coeffs["tm0"],
    )
    rhs_omega = f_omega_torch(
        delta_pred, omega_pred,
        coll_coeffs["vf0"], coll_coeffs["v"],
        coll_coeffs["a"], coll_coeffs["tm0"],
    ) / M_val
```

(continues on next page)

(continued from previous page)

```
    res_delta = ddelta_dt - rhs_delta
    res_omega = domega_dt - rhs_omega

    return torch.mean(res_delta ** 2) + torch.mean(res_omega ** 2)


PHYS_WEIGHT = 0.01
ADAM_EPOCHS = 5000
hist = {"total": [], "data": [], "phys": []}
t0 = time.time()

# ── Adam phase ──
opt = torch.optim.Adam(model.parameters(), lr=1e-3)
sched = torch.optim.lr_scheduler.CosineAnnealingLR(
    opt, T_max=ADAM_EPOCHS, eta_min=1e-5)

for ep in range(1, ADAM_EPOCHS + 1):
    opt.zero_grad()
    ld = torch.mean((model(X_u) - y_u) ** 2)
    lp = compute_physics_loss(model, X_f, coll_coeffs, M_val)
    loss = ld + PHYS_WEIGHT * lp
    loss.backward()
    opt.step()
    sched.step()
    hist["total"].append(loss.item())
    hist["data"].append(ld.item())
    hist["phys"].append(lp.item())

print(f"Adam {ADAM_EPOCHS} epochs:  "
      f"data={ld.item():.2e}  phys={lp.item():.2e}")

# ── L-BFGS refinement ──
lbfgs = torch.optim.LBFGS(
    model.parameters(), max_iter=1000,
    max_eval=1250, tolerance_grad=1e-9,
    tolerance_change=1e-11, history_size=100,
    line_search_fn="strong_wolfe")

lbfgs_it = [0]

def closure():
    lbfgs.zero_grad()
    ld = torch.mean((model(X_u) - y_u) ** 2)
    lp = compute_physics_loss(model, X_f, coll_coeffs, M_val)
    loss = ld + PHYS_WEIGHT * lp
    loss.backward()
    lbfgs_it[0] += 1
    hist["total"].append(loss.item())
    hist["data"].append(ld.item())
    hist["phys"].append(lp.item())
    return loss
```

(continues on next page)

(continued from previous page)

```
lbfgs.step(closure)

ld_f = torch.mean((model(X_u) - y_u) ** 2)
lp_f = compute_physics_loss(model, X_f, coll_coeffs, M_val)
print(f"L-BFGS {lbfgs_it[0]} iters: "
      f"data={ld_f.item():.2e}  phys={lp_f.item():.2e}")
print(f"Total training time: {time.time()-t0:.1f} s")
```

```
Adam 5000 epochs:  data=1.32e-05  phys=9.04e-04
L-BFGS 1030 iters: data=4.94e-07  phys=6.48e-04
Total training time: 260.9 s
```

Results

The trained PINN is evaluated against the ANDES TDS ground truth across the full (P_m, t) domain.

```
model.eval()

with torch.no_grad():
    X_t = torch.tensor(X_all, dtype=DTYPE, device=DEVICE)
    pred = model(X_t).cpu().numpy().reshape(
        usol_delta.shape[0], usol_delta.shape[1], 2)

pred_delta = pred[:, :, 0]
pred_omega = pred[:, :, 1]
err_delta = np.abs(pred_delta - usol_delta)
err_omega = np.abs(pred_omega - usol_omega)

print(f"delta:  MAE = {np.rad2deg(err_delta.mean()):.3f} deg,  "
      f"max error = {np.rad2deg(err_delta.max()):.3f} deg")
print(f"omega:  MAE = {err_omega.mean():.6f} pu,   "
      f"max error = {err_omega.max():.6f} pu")
```

```
delta:  MAE = 0.037 deg,  max error = 0.456 deg
omega:  MAE = 0.000063 pu,   max error = 0.001948 pu
```

```
fig = plt.figure(figsize=(14, 10))
gs = fig.add_gridspec(3, 3, hspace=0.40, wspace=0.30)

Pm_g2, T_g2 = np.meshgrid(Pm_vec, t_vec, indexing="ij")

# ▯▯ Row 1: surfaces and loss ▯▯
ax0 = fig.add_subplot(gs[0, 0])
c0 = ax0.pcolormesh(T_g2, Pm_g2, np.rad2deg(usol_delta),
                    shading="auto", cmap="RdBu_r")
ax0.set_xlabel("Time (s)")
ax0.set_ylabel("$P_m$ (pu)")
ax0.set_title("(a) ANDES TDS — $\\delta$")
```

(continues on next page)

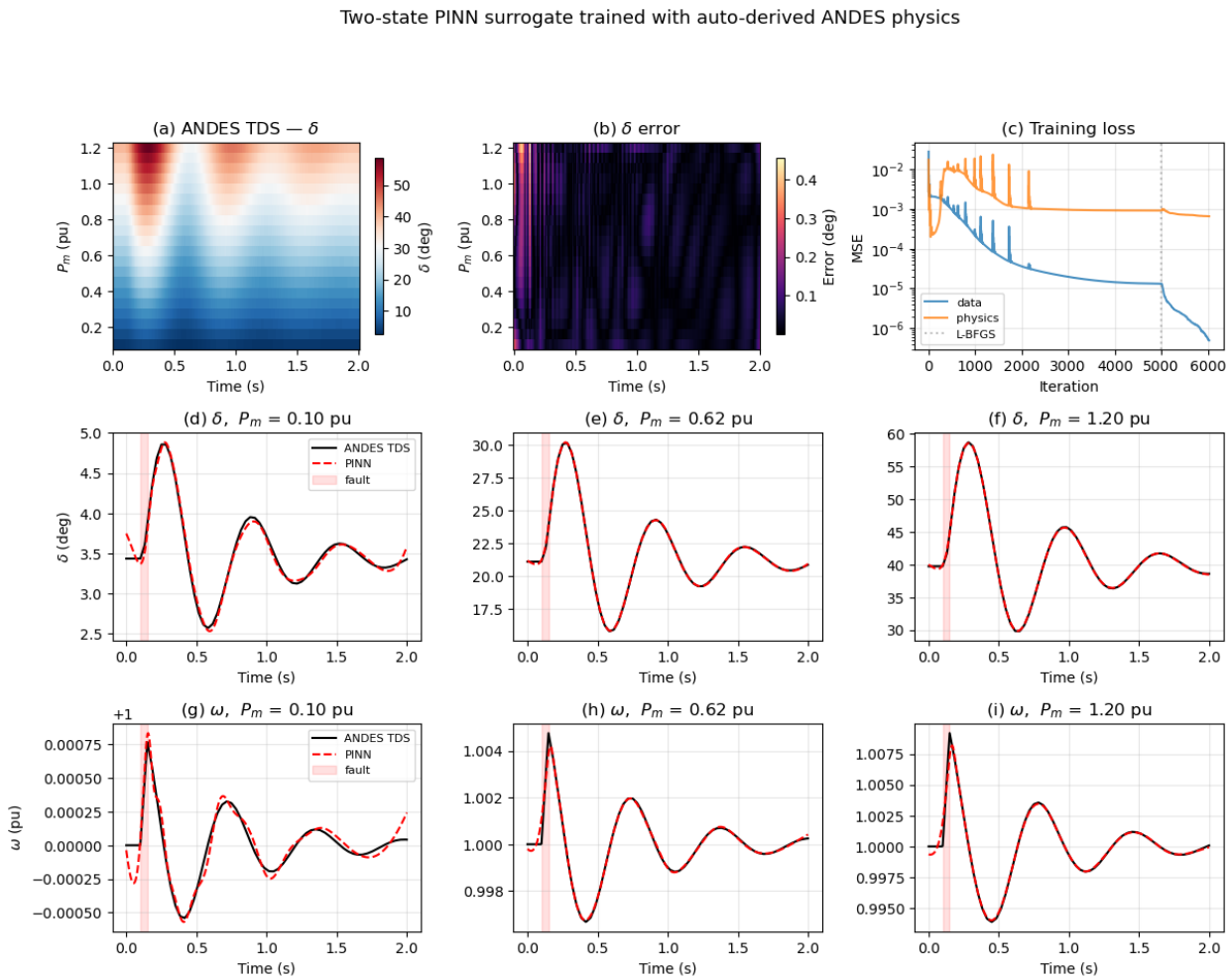
(continued from previous page)

```
fig.colorbar(c0, ax=ax0, label="$\\delta$ (deg)", shrink=0.85)

ax1 = fig.add_subplot(gs[0, 1])
c1 = ax1.pcolormesh(T_g2, Pm_g2, np.rad2deg(err_delta),
                    shading="auto", cmap="magma")
ax1.set_xlabel("Time (s)")
ax1.set_ylabel("$P_m$ (pu)")
ax1.set_title("(b) $\\delta$ error")
fig.colorbar(c1, ax=ax1, label="Error (deg)", shrink=0.85)

ax2 = fig.add_subplot(gs[0, 2])
ax2.semilogy(hist["data"], alpha=0.8, label="data", color="C0")
ax2.semilogy(hist["phys"], alpha=0.8, label="physics", color="C1")
ax2.axvline(ADAM_EPOCHS, color="gray", ls=":", alpha=0.5, label="L-BFGS")
ax2.set_xlabel("Iteration")
ax2.set_ylabel("MSE")
ax2.set_title("(c) Training loss")
ax2.legend(fontsize=8)
ax2.grid(True, alpha=0.3)

# ▰▰ Row 2: delta trajectory slices ▰▰
slice_idx = np.linspace(0, len(Pm_vec) - 1, 3, dtype=int)
for j, si in enumerate(slice_idx):
    ax = fig.add_subplot(gs[1, j])
    ax.plot(t_vec, np.rad2deg(usol_delta[si]), "k-", lw=1.5, label="ANDES TDS
↪")
    ax.plot(t_vec, np.rad2deg(pred_delta[si]), "r--", lw=1.5, label="PINN")
    ax.axvspan(0.1, 0.15, color="red", alpha=0.12,
               label="fault" if j == 0 else None)
    ax.set_xlabel("Time (s)")
    ax.set_title(f"({chr(100+j)}) $\\delta$,  $P_m$ = {Pm_vec[si]:.2f} pu")
    ax.grid(True, alpha=0.3)
    if j == 0:
        ax.set_ylabel("$\\delta$ (deg)")
        ax.legend(fontsize=8)

# ▰▰ Row 3: omega trajectory slices ▰▰
for j, si in enumerate(slice_idx):
    ax = fig.add_subplot(gs[2, j])
    ax.plot(t_vec, usol_omega[si], "k-", lw=1.5, label="ANDES TDS")
    ax.plot(t_vec, pred_omega[si], "r--", lw=1.5, label="PINN")
    ax.axvspan(0.1, 0.15, color="red", alpha=0.12,
               label="fault" if j == 0 else None)
    ax.set_xlabel("Time (s)")
    ax.set_title(f"({chr(103+j)}) $\\omega$,  $P_m$ = {Pm_vec[si]:.2f} pu")
    ax.grid(True, alpha=0.3)
    if j == 0:
        ax.set_ylabel("$\\omega$ (pu)")
        ax.legend(fontsize=8)

fig.suptitle(
    "Two-state PINN surrogate trained with auto-derived ANDES physics",
```

(continues on next page)

(continued from previous page)

```
    fontsize=13, y=1.01,
)
plt.show()
```



The two-state PINN achieves sub-degree accuracy for δ and sub-0.002 pu accuracy for ω across the full operating range, without any hand-coded physics. Both first-order ODEs from ANDES are enforced directly as the physics loss. The error is concentrated near the fault window where the ODE residuals are not enforced.

This pipeline generalizes to any ANDES model: calling `prepare()` on a higher-order machine (e.g., GENROU with multiple differential states) produces a larger set of symbolic equations that can be processed through the same elimination and lambdification steps.

Extend This Example

- **Higher-order models**: replace GENCLS with GENROU to capture sub-transient dynamics and flux-decay effects in the surrogate.
- **Larger networks**: parameterize the PINN over load levels or line impedances on multi-machine systems.
- **Fast contingency screening**: once trained, the surrogate evaluates orders of magnitude faster than TDS, enabling real-time stability assessment.
- **Reinforcement learning**: combine the surrogate with an RL agent for control design; see the *RL Oscillation Damping on SMIB* gallery example.

```
!andes misc -C
```

```
No output file found in the working directory.
```

3.1.5 Post-Fault Rotor Oscillation on an SMIB System

When a synchronous generator connected to an infinite bus experiences a transient fault, the rotor swings away from its equilibrium angle and then oscillates as the synchronizing torque pulls it back. The amplitude and persistence of these oscillations depend on the machine inertia, network strength, and the damping provided by flux dynamics and controllers.

This notebook constructs a single-machine infinite-bus (SMIB) system from scratch using the ANDES v2 API, applies a three-phase fault, and visualizes the resulting electromechanical oscillation. The system is intentionally configured with **zero mechanical damping** ($D = 0$) to produce a slowly-decaying swing that motivates the need for supplementary damping control — a task well suited to reinforcement learning, explored in a companion notebook.

```
%matplotlib inline

import numpy as np
from matplotlib import pyplot as plt

import andes

andes.config_logger(stream_level=30)
```

Build the SMIB System

The system has two buses connected by a single transmission line:

Bus	Role	Model
1	Generator bus	GENROU (6th-order salient-pole machine)
2	Infinite bus	GENCLS with very large inertia ($M = 6 \times 10^7$)

The generator is equipped with a TGOV1 governor and an EXST1 exciter. Mechanical damping is set to $D = 0$ so that all damping comes from the electromagnetic flux dynamics inherent in the GENROU model.

```
ss = andes.System(default_config=True)
ss.undill()

# --- network ---
ss.add("Bus", idx=1, name="Gen Bus", Vn=20.0, v0=1.05)
ss.add("Bus", idx=2, name="Inf Bus", Vn=20.0, v0=1.0)

ss.add("Line", idx=1, bus1=1, bus2=2, Vn1=20.0, Vn2=20.0,
       r=0.003, x=0.25, b=0.0)

# --- static generators (PFlow placeholders) ---
ss.add("PV",    idx=1, bus=1, Vn=20.0, p0=0.9, q0=0.0, v0=1.05)
ss.add("Slack", idx=2, bus=2, Vn=20.0, v0=1.0)

# --- infinite bus: very large GENCLS ---
ss.add("GENCLS", idx="INF", bus=2, gen=2, Sn=1e9, Vn=20.0,
       M=6e7, D=2e7, xd1=1e-9)

# --- finite machine: GENROU, D=0 ---
ss.add("GENROU", idx="GEN1", bus=1, gen=1, Sn=100, Vn=20.0,
       M=6.0, D=0.0,
       xd=1.8, xd1=0.3, xd2=0.25, Td10=8.0, Td20=0.03,
       xq=1.7, xq1=0.55, xq2=0.25, Tq10=0.4, Tq20=0.05,
       ra=0.003, xl=0.2)

# --- governor ---
ss.add("TGOV1", idx="TG1", syn="GEN1",
       T1=0.5, T2=2.0, T3=7.0, VMAX=99)

# --- exciter ---
ss.add("EXST1", idx="EX1", syn="GEN1", TR=0.02, KA=50.0)

# --- three-phase fault: 6 cycles on the generator bus ---
ss.add("Fault", idx="Flt1", bus=1, tf=1.0, tc=1.1, xf=1e-4)

ss.setup()
print(f"System: {ss.Bus.n} buses, {ss.Line.n} line, "
      f"{ss.GENROU.n} GENROU, {ss.GENCLS.n} GENCLS")
```

```
System: 2 buses, 1 line, 1 GENROU, 1 GENCLS
```

Power Flow

Solve for the initial steady-state operating point.

```
ss.PFlow.run()

print(f"Bus voltages:      {ss.Bus.v.v}")
print(f"Bus angles (deg):  {np.rad2deg(ss.Bus.a.v)}")
print(f"Generator P = {ss.PV.p0.v[0]:.3f} pu on {ss.config.mva:.0f} MVA base")
```

```
Bus voltages:      [1.05 1.  ]
Bus angles (deg):  [1.23240246e+01 5.59782271e-14]
Generator P = 0.900 pu on 100 MVA base
```

Apply a Three-Phase Fault and Run TDS

A solid three-phase fault ($X_f = 10^{-4}$ pu) is applied at the generator bus at $t = 1.0$ s and cleared at $t = 1.1$ s (6 cycles at 60 Hz). This drives the rotor angle far from equilibrium and excites the electromechanical oscillation mode.

```
ss.TDS.config.tf = 15.0
ss.TDS.run()
```

```
  0%|          | 0/100 [00:00<?, ?%/s]
```

```
True
```

Rotor Speed Response

The generator rotor speed ω deviates from nominal (1.0 pu) after the fault is cleared. With $D = 0$, the only source of damping is the electromagnetic interaction between the rotor flux linkages and the stator currents — an effect captured by the GENROU sub-transient model but absent in the classical (GENCLS) model.

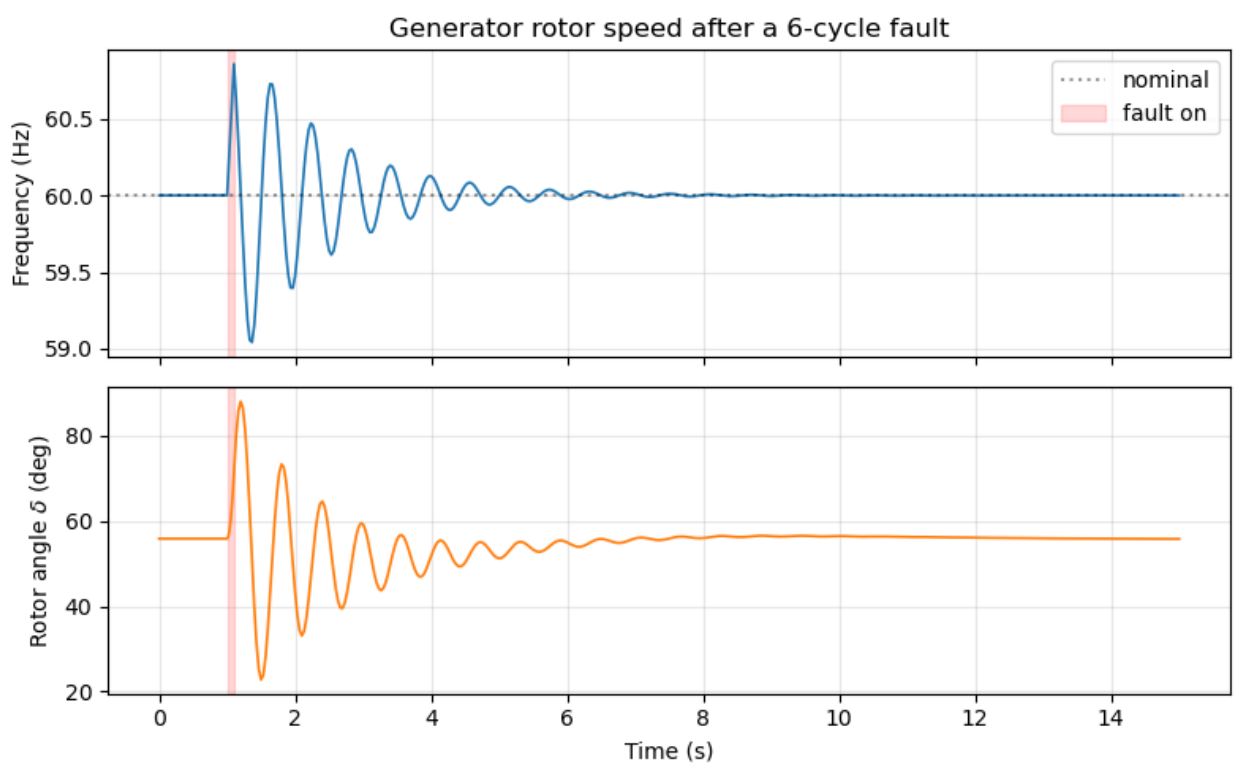
```
ts_omega = ss.TDS.get_timeseries(ss.GENROU.omega)
ts_delta = ss.TDS.get_timeseries(ss.GENROU.delta)

t = ts_omega.index.values
omega = ts_omega.values.flatten()
delta_deg = np.rad2deg(ts_delta.values.flatten())
freq_hz = omega * ss.config.freq        # convert to Hz
delta_omega = omega - 1.0               # deviation from nominal
```

```
fig, axes = plt.subplots(2, 1, figsize=(8, 5), sharex=True)

# --- rotor speed ---
ax = axes[0]
ax.plot(t, freq_hz, linewidth=1.2)
```

(continues on next page)

(continued from previous page)

```
ax.axhline(ss.config.freq, color="k", linestyle=":", alpha=0.4, label="nominal
↪")
ax.axvspan(1.0, 1.1, color="red", alpha=0.15, label="fault on")
ax.set_ylabel("Frequency (Hz)")
ax.legend(loc="upper right")
ax.set_title("Generator rotor speed after a 6-cycle fault")
ax.grid(True, alpha=0.3)

# --- rotor angle ---
ax = axes[1]
ax.plot(t, delta_deg, linewidth=1.2, color="C1")
ax.axvspan(1.0, 1.1, color="red", alpha=0.15)
ax.set_ylabel("Rotor angle $\\delta$ (deg)")
ax.set_xlabel("Time (s)")
ax.grid(True, alpha=0.3)

fig.tight_layout()
plt.show()
```



Quantify the Oscillation Decay

We measure the peak-to-peak amplitude of the ω deviation at successive swings. The ratio between consecutive peaks gives the per-cycle damping factor; the number of swings until the amplitude drops below a threshold quantifies the settling time.

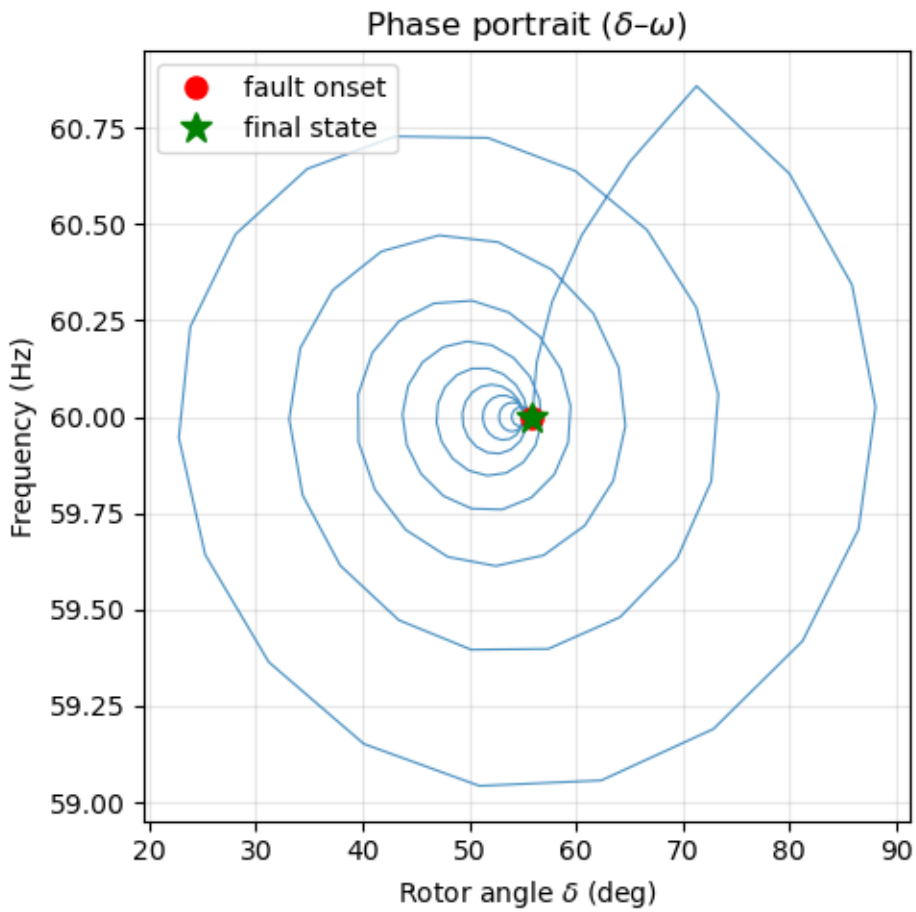
```
from scipy.signal import find_peaks

# analyse only the post-fault ring-down (t > 1.2 s)
mask = t > 1.2
t_pf, dev_pf = t[mask], np.abs(delta_omega[mask])

peaks, _ = find_peaks(dev_pf, distance=20)
peak_t = t_pf[peaks]
peak_amp = dev_pf[peaks]

# settling threshold: 0.1 % of nominal speed
threshold = 1e-3
settled_mask = peak_amp < threshold
if settled_mask.any():
    settle_idx = np.argmax(settled_mask)
    settle_time = peak_t[settle_idx] - 1.1   # seconds after fault clearing
    print(f"Settling time (|Δω| < {threshold}):  {settle_time:.1f} s "
          f"after fault clearing")
else:
    print("Oscillation has NOT settled within the simulation window.")

print(f"\nPeak-by-peak decay:")
print(f"{'Swing':>5s}  {'Time (s)':>8s}  {'|Δω| (pu)':>10s}  {'Decay ratio':>
↪12s}")
for i in range(min(8, len(peak_amp))):
    ratio = f"{peak_amp[i]/peak_amp[0]:.4f}" if i > 0 else "—"
    print(f"{i+1:5d}  {peak_t[i]:8.2f}  {peak_amp[i]:10.6f}  {ratio:>12s}")
```

```
Settling time (|Δω| < 0.001):  3.8 s after fault clearing

Peak-by-peak decay:
Swing  Time (s)   |Δω| (pu)   Decay ratio
    1      1.37    0.015958            —
    2      2.23    0.007851       0.4920
    3      3.10    0.004004       0.2509
    4      4.00    0.002096       0.1314
    5      4.87    0.000998       0.0625
    6      5.73    0.000624       0.0391
    7      6.93    0.000282       0.0177
    8      8.10    0.000126       0.0079
```

Phase Portrait

The (δ, ω) phase portrait reveals the trajectory of the rotor state. A damped oscillation spirals inward toward the equilibrium point; an undamped one would trace a closed orbit.

```
mask_ph = t >= 1.0  # from fault onset onward

fig, ax = plt.subplots(figsize=(5, 5))
ax.plot(delta_deg[mask_ph], freq_hz[mask_ph], linewidth=0.8, alpha=0.8)
ax.plot(delta_deg[mask_ph][0], freq_hz[mask_ph][0],
        "ro", markersize=8, label="fault onset")
ax.plot(delta_deg[mask_ph][-1], freq_hz[mask_ph][-1],
        "g*", markersize=12, label="final state")
ax.set_xlabel("Rotor angle $\\delta$ (deg)")
ax.set_ylabel("Frequency (Hz)")
ax.set_title("Phase portrait ($\\delta$-$\\omega$)")
ax.legend()
ax.grid(True, alpha=0.3)
fig.tight_layout()
plt.show()
```

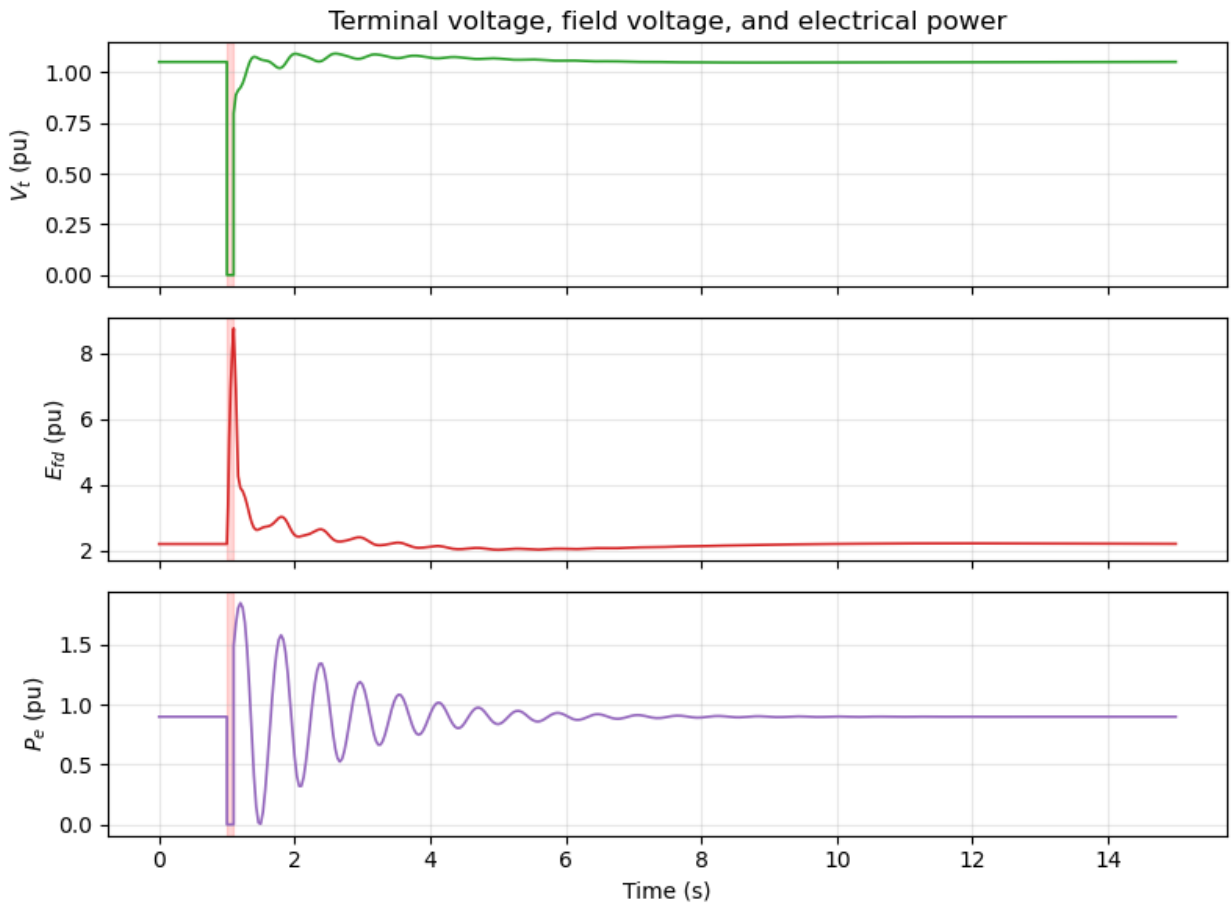


Terminal Voltage and Exciter Response

The exciter (EXST1) regulates the terminal voltage during and after the fault. Its action also contributes a small component of electrical damping torque.

```
ts_vt = ss.TDS.get_timeseries(ss.GENROU.v)      # terminal voltage
ts_vf = ss.TDS.get_timeseries(ss.GENROU.vf)     # field voltage (exciter␣
↪output)
ts_pe = ss.TDS.get_timeseries(ss.GENROU.Pe)      # electrical power
```

```
fig, axes = plt.subplots(3, 1, figsize=(8, 6), sharex=True)

ax = axes[0]
ax.plot(t, ts_vt.values.flatten(), linewidth=1.2, color="C2")
ax.axvspan(1.0, 1.1, color="red", alpha=0.15)
ax.set_ylabel("$V_t$ (pu)")
ax.set_title("Terminal voltage, field voltage, and electrical power")
ax.grid(True, alpha=0.3)

ax = axes[1]
ax.plot(t, ts_vf.values.flatten(), linewidth=1.2, color="C3")
ax.axvspan(1.0, 1.1, color="red", alpha=0.15)
ax.set_ylabel("$E_{fd}$ (pu)")
ax.grid(True, alpha=0.3)

ax = axes[2]
ax.plot(t, ts_pe.values.flatten(), linewidth=1.2, color="C4")
ax.axvspan(1.0, 1.1, color="red", alpha=0.15)
ax.set_ylabel("$P_e$ (pu)")
ax.set_xlabel("Time (s)")
ax.grid(True, alpha=0.3)

fig.tight_layout()
plt.show()
```



Toward Supplementary Damping Control

The 6-cycle fault on this SMIB system excites a ~1.1 Hz electromechanical oscillation mode. With $D = 0$ (no explicit mechanical damping), the oscillation takes roughly **4 seconds** to settle below 0.1% speed deviation, damped only by the GENROU flux dynamics and the exciter.

In larger interconnected systems, inter-area oscillation modes can be far less damped — and in some operating conditions, negatively damped (growing oscillations). This motivates supplementary damping controllers.

Below we use reinforcement learning to train a controller that injects auxiliary mechanical power p_{aux} through the governor to damp the oscillation faster, learning the optimal policy purely from simulation.

RL Environment Setup

ANDES provides `AndesEnv`, a Gymnasium-compatible environment that wraps the time-domain simulator. Key design choices:

Element	Choice	Rationale
Observation	$(\omega,\ \delta)$	Speed *and* angle give the agent full swing-equation state
Action	$p_{\text{aux}} \in [-0.1, 0.1]$ pu	Auxiliary power injected via the governor
Reward	$-(f_{\text{dev}}^2 + 0.1\, p_{\text{aux}}^2)$	Penalise frequency deviation (Hz) and control effort
Step interval	$\Delta t = 0.1$ s	~11 control actions per oscillation cycle
Episode	$t = 1.2$ s to 7.2 s	Start after fault clears, run 6 s of ring-down

Each `reset()` uses `TDS.reinit()` (~1 ms) instead of reloading the case from disk, enabling hundreds of episodes per minute.

```
from pathlib import Path
from andes.rl import AndesEnv
from stable_baselines3 import PPO
from stable_baselines3.common.callbacks import BaseCallback

CASE = str(Path("cases/smib/smib_rl.xlsx"))
```

```
def reward_fn(obs, action, env):
    """Penalise frequency deviation (in Hz) and control effort."""
    freq_dev_hz = (obs[0] - 1.0) * 60.0
    return -float(freq_dev_hz ** 2 + 0.1 * action[0] ** 2)


def disturbance_fn(env):
    """Fast-forward the simulation past the fault (cleared at t = 1.1 s)."""
    env.ss.TDS.config.tf = 1.2
    env.ss.TDS.run(no_summary=True)


env = AndesEnv(
    case=CASE,
    obs=[("GENROU", "omega"), ("GENROU", "delta")],
    acts=[("SynGen", "paux", ["GEN1"])],   # control only the finite machine
    reward_fn=reward_fn,
    dt=0.1,
    tf=7.2,
    disturbance_fn=disturbance_fn,
    action_low=-0.1,
    action_high=0.1,
)

print(f"Observation space: {env.observation_space}")
print(f"Action space:      {env.action_space}")
print(f"Steps per episode: {int((7.2 - 1.2) / 0.1)}")
```

```
Observation space: Box(-inf, inf, (2,), float32)
Action space:      Box(-0.1, 0.1, (1,), float32)
Steps per episode: 60
```

No-Control Baseline

Run one episode with zero p_{aux} to record the uncontrolled trajectory for comparison.

```
FREQ_NOM = env.ss.config.freq
zero_action = np.zeros(env.action_space.shape, dtype=np.float32)

obs, _ = env.reset(seed=42)
bl_t = [float(env.ss.dae.t)]
bl_omega = [obs[0]]
bl_reward = 0.0

while True:
    obs, r, term, trunc, info = env.step(zero_action)
    bl_t.append(info["t"])
    bl_omega.append(obs[0])
    bl_reward += r
    if term or trunc:
        break

bl_t = np.array(bl_t)
bl_omega = np.array(bl_omega)
print(f"Baseline total reward: {bl_reward:.2f}")
print(f"Frequency range: [{bl_omega.min()*FREQ_NOM:.2f}, "
      f"{bl_omega.max()*FREQ_NOM:.2f}] Hz")
```

```
Baseline total reward: -3.67
Frequency range: [59.16, 60.65] Hz
```

Train a PPO Agent

We use Stable-Baselines3 PPO with a small MLP policy (two hidden layers of 64 units). Training for 100 000 time steps takes a few minutes on a single CPU core.

```
class RewardLogger(BaseCallback):
    """Record per-episode total reward for the training curve."""
    def __init__(self):
        super().__init__()
        self.ep_rewards = []

    def _on_step(self):
        for info in self.locals.get("infos", []):
            if "episode" in info:
                self.ep_rewards.append(info["episode"]["r"])
```

(continues on next page)

(continued from previous page)

```
        return True


callback = RewardLogger()
```

```
model = PPO("MlpPolicy", env, n_steps=256, verbose=0)

model.learn(total_timesteps=100_000, callback=callback)
print(f"Training complete — {len(callback.ep_rewards)} episodes")
```

```
Training complete — 1668 episodes
```

```
/home/hcui9/miniforge3/envs/a/lib/python3.12/site-packages/stable_baselines3/
↪common/on_policy_algorithm.py:150: UserWarning: You are trying to run PPO␣
↪on the GPU, but it is primarily intended to run on the CPU when not using a␣
↪CNN policy (you are using ActorCriticPolicy which should be a MlpPolicy).␣
↪See https://github.com/DLR-RM/stable-baselines3/issues/1245 for more info.␣
↪You can pass `device='cpu'` or `export CUDA_VISIBLE_DEVICES=` to force␣
↪using the CPU.Note: The model will train, but the GPU utilization will be␣
↪poor and the training might take longer than on CPU.
  warnings.warn(
```

Evaluate the Trained Agent

Run one deterministic episode with the learned policy and record the trajectory alongside the baseline.

```
obs, _ = env.reset(seed=42)
ag_t = [float(env.ss.dae.t)]
ag_omega = [obs[0]]
ag_actions = []
ag_reward = 0.0

while True:
    action, _ = model.predict(obs, deterministic=True)
    obs, r, term, trunc, info = env.step(action)
    ag_t.append(info["t"])
    ag_omega.append(obs[0])
    ag_actions.append(action[0])
    ag_reward += r
    if term or trunc:
        break

ag_t = np.array(ag_t)
ag_omega = np.array(ag_omega)
ag_actions = np.array(ag_actions)

print(f"Agent total reward:    {ag_reward:.2f}")
print(f"Baseline total reward: {bl_reward:.2f}")
```

(continues on next page)

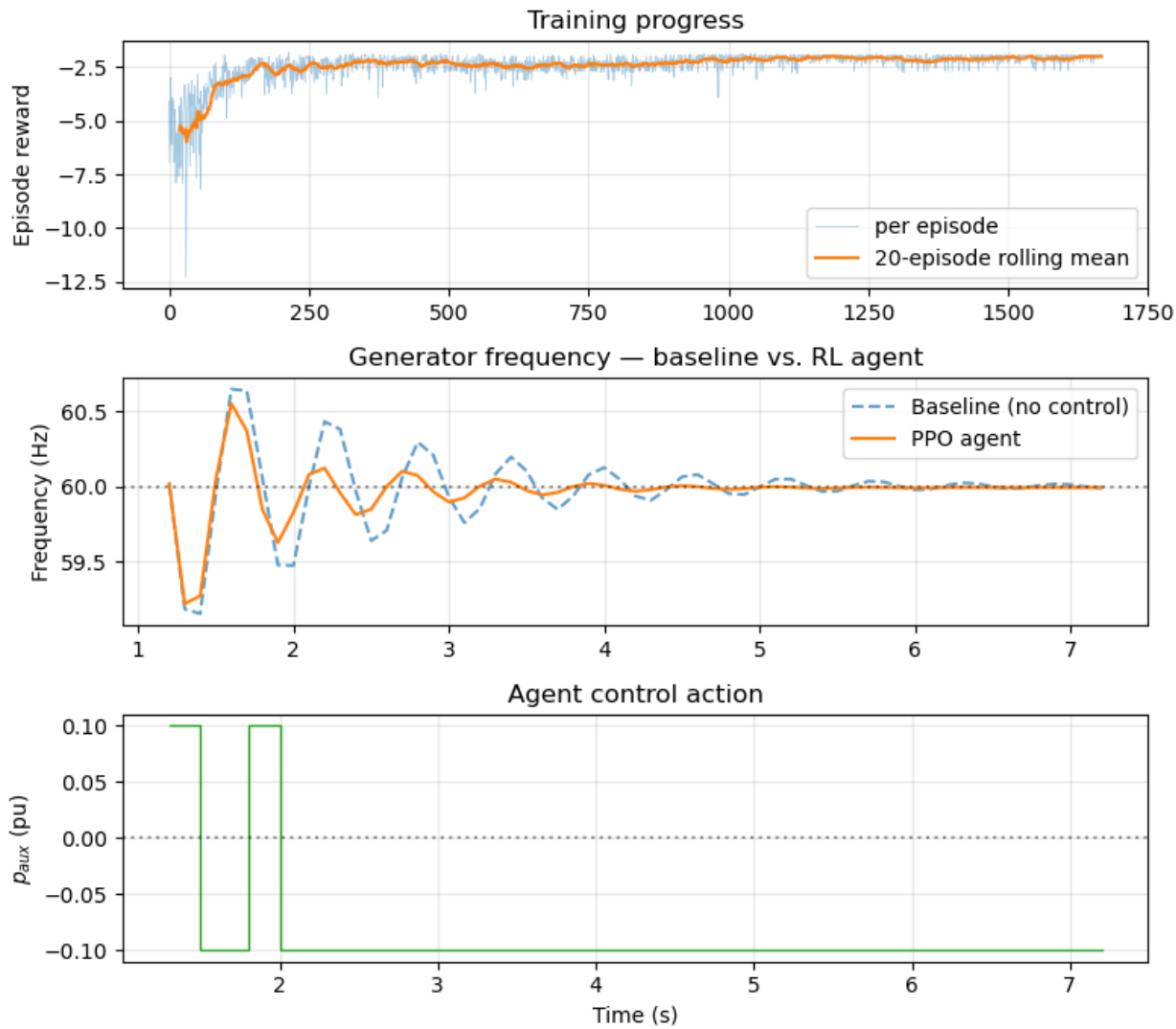
(continued from previous page)

```
print(f"Improvement:            {ag_reward - bl_reward:+.2f} "
      f"({(ag_reward - bl_reward) / abs(bl_reward) * 100:+.0f}%)")
```

```
Agent total reward:    -1.95
Baseline total reward: -3.67
Improvement:           +1.71 (+47%)
```

Results

```
fig, axes = plt.subplots(3, 1, figsize=(8, 7), sharex=False)

# (a) Training curve
ax = axes[0]
ax.plot(callback.ep_rewards, linewidth=0.5, alpha=0.4, label="per episode")
win = min(20, len(callback.ep_rewards))
if win > 1:
    rolling = np.convolve(callback.ep_rewards,
                          np.ones(win) / win, mode="valid")
    ax.plot(np.arange(win - 1, len(callback.ep_rewards)), rolling,
            linewidth=1.5, label=f"{win}-episode rolling mean")
ax.set_ylabel("Episode reward")
ax.set_title("Training progress")
ax.legend()
ax.grid(True, alpha=0.3)

# (b) Frequency: baseline vs agent
ax = axes[1]
ax.plot(bl_t, bl_omega * FREQ_NOM, "--", label="Baseline (no control)",
        linewidth=1.5, alpha=0.7)
ax.plot(ag_t, ag_omega * FREQ_NOM, "-", label="PPO agent",
        linewidth=1.5)
ax.axhline(FREQ_NOM, color="k", linestyle=":", alpha=0.4)
ax.set_ylabel("Frequency (Hz)")
ax.set_title("Generator frequency — baseline vs. RL agent")
ax.legend()
ax.grid(True, alpha=0.3)

# (c) Agent actions
ax = axes[2]
ax.step(ag_t[1:], ag_actions, where="post", linewidth=1.0, color="C2")
ax.axhline(0, color="k", linestyle=":", alpha=0.4)
ax.set_ylabel("$p_{aux}$ (pu)")
ax.set_xlabel("Time (s)")
ax.set_title("Agent control action")
ax.grid(True, alpha=0.3)

fig.tight_layout()
plt.show()
```



Extend This Example

- **Increase training**: 20 000 steps is a minimal run; 100 000+ steps with hyperparameter tuning will yield a stronger policy.
- **Richer observation**: add terminal voltage V_t or electrical power P_e to give the agent more information about the system state.
- **Multi-machine systems**: replace the SMIB with the IEEE 14-bus or 39-bus system to learn coordinated damping across multiple generators.
- **Compare with classical PSS**: add an IEEEST stabiliser and overlay its response to see whether RL matches classical control design.

CHAPTER

FOUR

MODELING

This section provides an in-depth explanation of how the ANDES framework is set up for symbolic modeling and numerical simulation. It aims to help developers understand the internal architecture and implement customized DAE models.

4.1 Who This Is For

This section is for developers who want to:

- Understand how ANDES models work internally
- Implement new power system device models
- Extend ANDES capabilities with custom components

For users who want to:

- Run simulations: see the *Tutorials* section
- Inspect model equations: see the *Inspecting Model Equations* tutorial

4.2 Modeling Concepts

Foundational concepts for understanding the ANDES modeling framework.

4.2.1 Key Ideas

Hybrid Symbolic-Numeric Framework

ANDES uses SymPy to define model equations symbolically, then generates optimized numerical code automatically. This enables:

- Automatic Jacobian derivation
- Self-documenting models with LaTeX equations
- Rapid prototyping of new models

Atomic Types

Models are built from three fundamental atom types: parameters, variables, and services. These share a common **v-provider** interface (the `v` attribute) that enables interoperability in equations. Variables additionally serve as **e-providers**, contributing equation residuals to the DAE system.

DAE Formulation

Power system dynamics are modeled as differential-algebraic equations:

- **Differential equations** (f): Generator dynamics, controller states
- **Algebraic equations** (g): Network power balance, algebraic constraints

System Architecture

The `System` class orchestrates models, routines, and the DAE arrays.

4.2.2 Hybrid Symbolic-Numeric Framework

ANDES uses a unique hybrid symbolic-numeric framework that combines the expressiveness of symbolic mathematics with the efficiency of numerical computation.

Key Concept

Instead of writing numerical code directly, you **describe models using equations and building blocks**. ANDES then:

1. Parses symbolic equation strings
2. Computes partial derivatives automatically
3. Generates optimized numerical code
4. Executes vectorized calculations

Benefits

For Model Developers

- **Write equations, not code**: Define differential equations as strings like `'omega - 1'`
- **Automatic Jacobians**: No need to derive and code partial derivatives
- **Modular blocks**: Reuse transfer functions (Lag, LeadLag, PIController)
- **Built-in validation**: Framework catches common modeling errors

For Users

- **Identical models**: PSS/E parameters work directly without conversion
- **Fast simulation**: Generated code is optimized and vectorized
- **Extensible**: Add new models without modifying core solver

Framework Components

```
┌──────────────────────────────────────────────┐
│                  Model Definition            │
│   Parameters  Variables  Equations  Blocks   │
└──────────────────────────────────────────────┘
                       │
                       ▼
┌──────────────────────────────────────────────┐
│             Symbolic Processing              │
│   Parse → Derive → Optimize → Generate       │
└──────────────────────────────────────────────┘
                       │
                       ▼
┌──────────────────────────────────────────────┐
│             Numerical Simulation             │
│   Initialize → Solve → Iterate → Output      │
└──────────────────────────────────────────────┘
```

Code Generation

When you first run ANDES or modify models, code generation occurs:

```
# Triggered automatically, or manually with:
andes prepare
```

This creates optimized Python code stored in `~/.andes/pycode/`. Generation runs once unless models change.

What Gets Generated

- Equation evaluation functions
- Jacobian matrix builders
- Initialization routines
- Variable indexing maps

Example: Model Definition

A simplified turbine governor in ANDES:

```
class TGOV1(TGOVData, Model):
    def __init__(self, system, config):
        TGOVData.__init__(self)
        Model.__init__(self, system, config)

        # Services (intermediate constants)
        self.gain = ConstService(v_str='u/R')

        # Variables with equation strings
        self.pout = Algeb(
            v_str='tm0',
            e_str='(paux + pref + gain * (omega - 1) - pout)'
        )

        # Transfer function block
        self.LG = Lag(u=self.pout, T=self.T1, K=1)
```

Key observations:

- `v_str` defines initial value calculation
- `e_str` defines the algebraic equation (set to zero)
- `Lag` block encapsulates first-order dynamics
- No numerical code for derivatives—handled automatically

Symbolic Variables

Within equation strings, you can reference:

Type	Example	Description
Parameters	`R`, `T1`	Model parameters
Variables	`omega`, `pout`	State/algebraic variables
External	`omega`	Variables from other models
Services	`gain`	Intermediate calculations
Block outputs	`LG_y`	Output of transfer blocks

Integration with Solver

During simulation:

1. **System** collects all model equations
2. **Equations** are evaluated using generated code
3. **Jacobians** are built incrementally by model
4. **Solver** finds the solution iteratively
5. **Results** are distributed back to models

See Also

- *Atomic Types* - v-provider and e-provider concepts
- *System Architecture* - System class internals
- *DAE Formulation* - DAE mathematical background
- *Parameters* - Parameter types
- *Variables* - Variable types

4.2.3 Atomic Types

ANDES contains three fundamental types of atom classes for building DAE models: **parameters**, **variables**, and **services**. These atom classes share common interfaces that enable interoperability and composability when defining model equations.

Value Provider (v-provider)

A **value provider** class (or *v-provider* for short) is any class with a member attribute named `v`, which should be a list or a 1-dimensional NumPy array of values. The `v` attribute contains the numerical values that will be substituted into equations during computation.

Note: All atom classes are v-providers. Every parameter, variable, and service instance must contain values accessible through the `v` attribute.

How v-providers Work

When you define parameters in a model:

```
self.v0 = NumParam(default=1.0, info='Initial voltage')
self.b = NumParam(default=10.0, info='Susceptance')

# At runtime, these might contain:
# self.v0.v = np.array([1.0, 1.05, 1.1])
# self.b.v  = np.array([10., 10., 10.])
```

When these parameters are used in an equation string:

```
self.v = ExtAlgeb(model='Bus', src='v',
                  indexer=self.bus,
                  e_str='v0 ** 2 * b')
```

During equation evaluation, `v0` and `b` are substituted with their respective `v` arrays (`self.v0.v` and `self.b.v`). The symbolic expression `v0 ** 2 * b` becomes a vectorized numerical computation.

Interoperability Through the v Interface

The shared `v` interface enables seamless substitution between different atom types. Consider this parameter definition:

```
self.v0 = NumParam(default=1.0)
```

You could replace it with a service that computes the value:

```
self.v0 = ConstService(v_str='1.0')
```

Equations using `v0` continue to work without modification, because both `NumParam` and `ConstService` provide values through the same `v` attribute. This design pattern allows:

- Parameters to be replaced by computed services
- Services to reference other services or parameters
- Variables to be used in place of parameters in equation strings

Equation Provider (e-provider)

An **equation provider** class (or *e-provider*) is any class with a member attribute named `e`, which should be a 1-dimensional array. The `e` array stores equation evaluation results that are accumulated into the numerical DAE system at addresses specified by the `a` attribute.

Note: Currently, only variables (differential and algebraic) are e-providers. Parameters and services provide values but do not contribute equations to the DAE.

How e-providers Work

When you define an external variable that links to bus voltage:

```
self.v = ExtAlgeb(model='Bus', src='v',
                  indexer=self.bus,
                  e_str='v0 ** 2 * b')
```

Three things happen:

1. **Address retrieval**: The addresses of the corresponding voltage variables are retrieved into `self.v.a`
2. **Equation evaluation**: The expression `v0 ** 2 * b` is evaluated and stored in `self.v.e`
3. **DAE assembly**: Values in `self.v.e` are summed into the system DAE array at the addresses in `self.v.a`

The v-e-a Triad for Variables

Variables have three key attributes that work together:

Attribute	Purpose	Description
`v`	Values	Current values of the variable (solution)
`e`	Equations	Equation contributions (residuals)
`a`	Addresses	Indices into the system DAE arrays

This triad enables the numerical integration scheme:

- **`v`** holds the current state/algebraic values being solved
- **`e`** accumulates equation right-hand-sides from this model
- **`a`** specifies where in the global DAE this variable lives

Summary

Atom Type	v-provider	e-provider	Purpose
Parameter	Yes	No	Input data for models
Service	Yes	No	Intermediate calculations
Variable	Yes	Yes	DAE unknowns and equations

The v-provider interface creates a uniform way to access values, enabling:

- Symbolic equation processing that works with any value source
- Easy substitution between parameters, services, and variables
- Clean separation between data definition and equation evaluation

See Also

- *Parameters* - Parameter types and usage
- *Variables* - Variable types and the v-e-a attributes
- *Services* - Service types for intermediate calculations

4.2.4 System Architecture

The `System` class is the central orchestrator in ANDES, managing models, routines, and DAE data structures. The full API reference is found at `andes.system.System`.

System Overview

```
import andes

ss = andes.load('case.xlsx')
# ss is a System instance containing:
# - All loaded models (ss.Bus, ss.PQ, ss.GENROU, etc.)
# - Analysis routines (ss.PFlow, ss.TDS, ss.EIG)
# - DAE storage (ss.dae)
# - Configuration (ss.config)
```

Dynamic Imports

System dynamically imports groups, models, and routines at creation. To add new models, groups or routines, edit the corresponding registration file following existing examples.

```
System
├── Groups (StaticLoad, Generator, Exciter, ...)
├── Models (Bus, PQ, GENROU, ESST1A, ...)
└── Routines (PFlow, TDS, EIG)
```

`andes.system.System.`**`import_models`**(*self*)

Delegate to `andes.system.registry.RegistryLoader`.

`andes.system.System.`**`import_groups`**(*self*)

Delegate to `andes.system.registry.RegistryLoader`.

`andes.system.System.`**`import_routines`**(*self*)

Delegate to `andes.system.registry.RegistryLoader`.

Code Generation

Under the hood, all models whose equations are provided in strings need to be processed to generate executable functions for simulations. We call this process "code generation". Code generation utilizes SymPy and can take up to one minute.

Code generation is automatically triggered upon the first ANDES run or whenever model changes are detected. The generated code is stored and reused for speed up.

The generated Python code is called `pycode`. It is a Python package (folder) with each module (a `.py` file) storing the executable Python code and metadata for numerical simulation. The default storage path is `~/.andes/`.

Note: Code generation has been done if you have executed `andes`, `andes selftest`, or `andes prepare`.

Warning: For developers: when models are modified (such as adding new models or changing equation strings), code generation needs to be executed again for consistency. ANDES can automatically detect changes, and it can be manually triggered from command line using `andes prepare -i`.

`andes.system.System.`**`prepare`**(*self*, *quick=False*, *incremental=False*, *models=None*, *nomp=False*, *ncpu=2*)

Delegate to *andes.system.codegen.CodegenManager*.

`andes.system.System.`**`undill`**(*self*, *autogen_stale=True*)

Delegate to *andes.system.codegen.CodegenManager*.

DAE Storage

The numerical DAE arrays are stored in `System.dae`:

```
ss.dae.x    # State variables (differential)
ss.dae.y    # Algebraic variables
ss.dae.f    # Differential equations (dx/dt)
ss.dae.g    # Algebraic equations (= 0)
ss.dae.t    # Current time
```

Time Series

After TDS, use `get_timeseries()` to extract results as a pandas DataFrame:

```
omega_df = ss.TDS.get_timeseries(ss.GENROU.omega)  # State variable
voltage_df = ss.TDS.get_timeseries(ss.Bus.v)       # Algebraic variable
```

The underlying arrays are also available for direct access:

```
ss.dae.ts.t    # Time points
ss.dae.ts.x    # State history (n_steps x n_states)
ss.dae.ts.y    # Algebraic history (n_steps x n_algebs)
```

class andes.variables.dae.**DAE**(*system*)

Class for storing numerical values of the DAE system, including variables, equations and first order derivatives (Jacobian matrices).

Variable values and equation values are stored as `numpy.ndarray`, while Jacobians are stored as `kvxopt.spmatrix`. The defined arrays and descriptions are as follows:

DAE Array	Description
x	Array for state variable values
y	Array for algebraic variable values
z	Array for 0/1 limiter states (if enabled)
f	Array for differential equation derivatives
Tf	Left-hand side time constant array for f
g	Array for algebraic equation mismatches

The defined scalar member attributes to store array sizes are

Scalar	Description
m	The number of algebraic variables/equations
n	The number of state variables/equations
o	The number of limiter state flags

The derivatives of f and g with respect to x and y are stored in four `kvxopt.spmatrix` sparse matrices: **fx**, **fy**, **gx**, and **gy**, where the first letter is the equation name, and the second letter is the variable name.

Notes

DAE in ANDES is defined in the form of

$$
\begin{aligned}
T\dot{x} &= f(x, y) \\
0 &= g(x, y)
\end{aligned}
$$

DAE does not keep track of the association of variable and address. Only a variable instance keeps track of its addresses.

Device Connectivity

Use `find_connected()` to find all devices referencing a given bus or device:

```
ss.find_connected('Bus', 1)
# OrderedDict([('Slack', [1]), ('Line', ['Line_1', 'Line_2']),
#              ('GENROU', ['GENROU_1']), ('BusFreq', ['BusFreq_2'])])
```

This works for any model or group, not just buses:

```
ss.find_connected('SynGen', 'GENROU_1')
# OrderedDict([('TGOV1', ['TGOV1_1']), ('ESST3A', ['ESST3A_2'])])
```

`andes.system.System.`**`find_connected`**(*self*, *model_or_group*, *idx*)

Find all devices connected to a given device.

Scans `IdxParam` references across all models to find devices that point to the specified target device.

Parameters

model_or_group
[str] Model name (e.g., `'Bus'`) or group name (e.g., `'ACNode'`).

idx
[str, int, float] Device idx to query.

Returns

OrderedDict
`{model_name: [idx, ...]}` for each model with at least one device referencing the target. Empty models are omitted.

Examples

Find all devices connected to Bus 1:

```
ss.find_connected('Bus', 1)
# OrderedDict([('PQ', [1]), ('Line', [1, 4]), ('GENROU', [1])])
```

Decentralized Architecture

ANDES uses a decentralized architecture between models and DAE value arrays. In this architecture, variables are initialized and equations are evaluated inside each model. Then, `System` provides methods for collecting initial values and equation values into `DAE`, as well as copying solved values back to each model.

1. **Models own their data**: Each model has local copies of variable values
2. **System orchestrates**: Collects/distributes values between models and DAE
3. **Parallel-friendly**: Model equations can be evaluated independently

Data Flow

```
┌─────────┐          ┌─────────┐          ┌─────────┐
│ Model   │ ──────►  │   DAE   │  ◄──────  │ Model   │
│ (PQ)    │          │ arrays  │          │ (Gen)   │
└─────────┘          └─────────┘          └─────────┘
      │                   ▲                    │
      │                   │                    │
      └───────────────────┼────────────────────┘
                          │
                          │
                     ┌────┴────┐
                     │ System  │
                     │ methods │
                     └─────────┘
```

The collection of values from models needs to follow protocols to avoid conflicts. Details are given in the *Variables* section.

`andes.system.System.`**`vars_to_dae`**(*self*, *model*)

Copy variables values from models to *System.dae*.

This function clears *DAE.x* and *DAE.y* and collects values from models.

`andes.system.System.`**`vars_to_models`**(*self*)

Copy variable values from *System.dae* to models.

`andes.system.System.`**`_e_to_dae`**(*self*, *eq_name: str | Tuple = ('f', 'g')*)

Helper function for collecting equation values into *System.dae.f* and *System.dae.g*.

Parameters

eq_name
['x' or 'y' or tuple] Equation type name

Jacobian Matrices

System builds sparse Jacobian matrices incrementally:

```
ss.dae.fx    # df/dx (differential w.r.t. states)
ss.dae.fy    # df/dy (differential w.r.t. algebraic)
ss.dae.gx    # dg/dx (algebraic w.r.t. states)
ss.dae.gy    # dg/dy (algebraic w.r.t. algebraic)
```

Matrix Sparsity Patterns

The largest overhead in building and solving nonlinear equations is the building of Jacobian matrices. This is especially relevant when using the implicit integration approach which algebraizes the differential equations. Given the unique data structure of power system models, the sparse matrices for Jacobians are built **incrementally**, model after model.

There are two common approaches to incrementally build a sparse matrix:

Approach 1: In-place addition

```
self.fx += spmatrix(v, i, j, (n, n), 'd')
```

Although simple, this involves creating and discarding temporary objects on the right hand side and, worse, changing the sparse pattern of `self.fx`.

Approach 2: Collect and construct Store the rows, columns and values in array-like objects and construct the Jacobians at the end. Very efficient but does not allow accessing the sparse matrix while building.

ANDES approach: Pre-allocation ANDES uses a pre-allocation approach to avoid changing sparse patterns by filling values into a known sparse matrix pattern. System collects the indices of rows and columns for each Jacobian matrix. Before in-place additions, ANDES builds a temporary zero-filled `spmatrix`, to which the actual Jacobian values are written later. Since these in-place add operations only modify existing values, it does not change the pattern and thus avoids memory copying.

`andes.system.System.`**`store_sparse_pattern`**(*self*, *models: OrderedDict*)

Collect and store the sparsity pattern of Jacobian matrices.

This is a runtime function specific to cases.

Notes

For *gy* matrix, always make sure the diagonal is reserved. It is a safeguard if the modeling user omitted the diagonal term in the equations.

Calling Model Methods

System is an orchestrator for calling shared methods of models. These API methods are defined for initialization, equation update, Jacobian update, and discrete flags update.

The following methods take an argument `models`, which should be an `OrderedDict` of models with names as keys and instances as values.

Initialization

`andes.system.System.`**`init`**(*self*, *models: OrderedDict*, *routine: str*)

Initialize the variables for each of the specified models.

For each model, the initialization procedure is:

- Get values for all *ExtService*.
- Call the model *init()* method, which initializes internal variables.
- Copy variables to DAE and then back to the model.

Equation Updates

`andes.system.System.`**`e_clear`**(*self*, *models: OrderedDict*)

Clear equation arrays in DAE and model variables.

This step must be called before calling *f_update* or *g_update* to flush existing values.

`andes.system.System.`**`l_update_var`**(*self*, *models: OrderedDict*, *niter=0*, *err=None*)

Update variable-based limiter discrete states by calling `l_update_var` of models.

This function is must be called before any equation evaluation.

`andes.system.System.`**`f_update`**(*self*, *models: OrderedDict*)

Call the differential equation update method for models in sequence.

Notes

Updated equation values remain in models and have not been collected into DAE at the end of this step.

`andes.system.System.`**`g_update`**(*self*, *models: OrderedDict*)

Call the algebraic equation update method for models in sequence.

Notes

Like *f_update*, updated values have not collected into DAE at the end of the step.

`andes.system.System.l_update_eq`(*self*, *models: OrderedDict*, *init=False*, *niter=0*)

Update equation-dependent limiter discrete components by calling `l_check_eq` of models. Force set equations after evaluating equations.

This function is must be called after differential equation updates.

`andes.system.System.j_update`(*self*, *models: OrderedDict*, *info=None*)

Call the Jacobian update method for models in sequence.

The procedure is - Restore the sparsity pattern with `andes.variables.dae.DAE.restore_sparse()` - For each sparse matrix in (fx, fy, gx, gy), evaluate the Jacobian function calls and add values.

Notes

Updated Jacobians are immediately reflected in the DAE sparse matrices (fx, fy, gx, gy).

Model Calling Protocol

System calls model methods in a defined order during each iteration:

1. `model.init()` - Initialize variables
2. `model.l_update_var()` - Update discrete flags (pre-equation)
3. `model.f_update()` - Evaluate differential equations
4. `model.g_update()` - Evaluate algebraic equations
5. `model.j_update()` - Update Jacobian contributions
6. `model.l_update_eq()` - Update discrete flags (post-equation)

External Variable Protocol

When models share variables (e.g., Bus voltage accessed by loads):

Flag	Purpose
`v_setter=False`	Values at same address are summed
`v_setter=True`	This variable sets the final value
`e_setter=False`	Equation values are summed
`e_setter=True`	This equation sets the final value

Example: PV generator sets bus voltage initial value:

```
# In PV model
self.v = ExtAlgeb(src='v', model='Bus',
                  indexer=self.bus,
                  v_str='v0',
                  v_setter=True)  # Overwrite bus voltage
```

Configuration

System, models and routines have a member attribute `config` for specific configurations. System manages all configs, including saving to a config file and loading back.

```
ss.config               # System config
ss.PFlow.config         # Power flow config
ss.TDS.config           # TDS config
ss.GENROU.config        # Model-specific config
```

`andes.system.System.`**`save_config`**(*self*, *file_path=None*, *overwrite=False*)

Delegate to `andes.system.config_runtime.SystemConfigRuntime`.

> **Warning:** Configs from files are passed to *model constructors* during instantiation. If you need to modify config for a run, it must be done before instantiating `System`, or before running `andes` from command line. Directly modifying `Model.config` may not take effect or have side effects in the current implementation.

Device Lifecycle

Understanding how devices are created and managed helps when extending cases programmatically or developing new models.

Data Loading Pipeline

When loading from files (xlsx, JSON, PSS/E), ANDES follows this sequence:

1. **File parsing**: I/O readers (`andes/io/xlsx.py`, `json.py`, `psse.py`) parse the input format into dictionaries
2. **Device registration**: Each row calls `System.add(model_name, param_dict)`:
   - Validates the model exists
   - Gets a unique `idx` from the device's group
   - Calls `Model.add()` to store parameter values
   - Registers the device with its group
3. **System setup**: `System.setup()` finalizes the structure (addresses, code generation)

```
# This is what happens internally during andes.load():
# (simplified from andes/io/xlsx.py)
for name, df in df_models.items():
    for row in df.to_dict(orient='records'):
        system.add(name, row)  # Called once per device row
```

`andes.system.System.add`(*self*, *model_name*, *param_dict=None*, ***kwargs*)

Add a device instance for an existing model.

This method calls the `add` method of the model and registers the device *idx* to its group.

Parameters can be passed as a dictionary, as keyword arguments, or both. When both are provided, keyword arguments are merged into the dictionary (kwargs take precedence on conflicts).

Parameters

model_name
[str] Name of the model (e.g., `'Fault'`, `'Toggle'`, `'PQ'`).

param_dict
[dict, optional] Dictionary of parameter names to values.

****kwargs**
Parameter names and values as keyword arguments.

Returns

idx
The assigned device index.

Examples

Keyword arguments are the preferred style:

```
ss.add('Fault', bus=5, tf=1.0, tc=1.1)
```

Models with a `model` parameter (e.g., `Alter`, `Toggle`) can now use keyword arguments directly:

```
ss.add('Alter', model='TGOV1', dev=1, src='paux0',
       t=1.0, method='=', amount=0.05)
ss.add('Toggle', model='Line', dev='Line_5', t=1.0)
```

Programmatic Device Addition

Devices can be added programmatically before `setup()`:

```
ss = andes.load('case.xlsx', setup=False)
ss.add('Fault', bus=3, tf=1.0, tc=1.1)
ss.setup()
```

Key constraints:

- Devices cannot be added after `setup()` (raises `NotImplementedError`)
- Referenced devices must exist (e.g., `bus` must reference a valid `Bus.idx`)
- See *Parameters* for mandatory vs. optional parameters and per-unit conventions

Timed Event Mechanism

Disturbance devices (`Fault`, `Toggle`, `Alter`) use `TimerParam` to schedule callbacks during time-domain simulation. These models belong to the `TimedEvent` group.

How It Works

1. `TimerParam` stores a time value and a callback function
2. During TDS, the solver checks if `dae.t` matches any timer values
3. When triggered, the callback executes (e.g., `Fault.apply_fault()`)
4. The callback modifies system state (shunt admittance, device `u` flag, parameter values)

```
# From andes/models/timer.py - Fault implementation
class Fault(ModelData, Model):
    def __init__(self, system, config):
        # ...
        self.tf = TimerParam(info='Bus fault start time',
                             callback=self.apply_fault)
        self.tc = TimerParam(info='Bus fault end time',
                             callback=self.clear_fault)

    def apply_fault(self, is_time: np.ndarray):
        """Apply fault when t = tf."""
        for i in range(self.n):
            if is_time[i] and self.u.v[i]:
                self.uf.v[i] = 1  # Enable fault equations
                # Store pre-fault algebraic variables for restoration
        return action
```

The fault equations inject a shunt admittance at the faulted bus:

```
self.a = ExtAlgeb(model='Bus', src='a', indexer=self.bus,
                  e_str='u * uf * (v ** 2 * gf)')  # Active power
self.v = ExtAlgeb(model='Bus', src='v', indexer=self.bus,
                  e_str='-u * uf * (v ** 2 * bf)')  # Reactive power
```

Available Timed Event Models

Model	Purpose	Key Parameters
`Fault`	Three-phase fault at a bus	`bus`, `tf`, `tc`, `xf`, `rf`
`Toggle`	Switch device on/off	`model`, `dev`, `t`
`Alter`	Modify parameter value at runtime	`model`, `dev`, `src`, `t`, `method`, `amount`

- **Fault**: Applies a shunt impedance (`xf`, `rf`) to ground at time `tf`, clears at `tc`
- **Toggle**: Negates the `u` (connectivity) field of any device at time `t`
- **Alter**: Applies arithmetic operations (`+`, `-`, `*`, `/`, `=`) to any parameter or service

For practical examples of adding disturbances, see *Time-Domain Simulation*.

See Also

- *Hybrid Symbolic-Numeric Framework* - Symbolic-numeric framework
- *Atomic Types* - Value and equation providers
- *DAE Formulation* - DAE mathematical details

4.2.5 DAE Formulation

ANDES models power systems as a set of Differential-Algebraic Equations (DAEs).

Mathematical Form

The general DAE system is:

$$\mathbf{M}\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{y})$$

$$\mathbf{0} = \mathbf{g}(\mathbf{x}, \mathbf{y})$$

Where:

- $\mathbf{x}$ are differential (state) variables
- $\mathbf{y}$ are algebraic variables
- $\mathbf{M}$ is the mass matrix (often identity)
- $\mathbf{f}$ are differential equations
- $\mathbf{g}$ are algebraic equations

Variable Types

State Variables ($\mathbf{x}$)

Describe dynamic behavior—change continuously over time.

Examples:

- Generator rotor angle (δ)
- Generator speed deviation ($\omega - 1$)
- Exciter states
- Governor states

```
# In model definition
self.omega = State(v_str='1.0',
                   e_str='(Tm - Te - D*(omega-1)) / M')
```

Algebraic Variables ($\mathbf{y}$)

Satisfy instantaneous constraints—can change discontinuously.

Examples:

- Bus voltage magnitude (V)
- Bus voltage angle (θ)
- Generator electrical power (P_e)
- Current injections

```
# In model definition
self.v = Algeb(v_str='1.0',
               e_str='P - v * sum(Y * V * cos(theta))')
```

Power Flow vs Dynamic

Power Flow (Steady-State)

All derivatives are zero: $\dot{\mathbf{x}} = 0$

The system becomes purely algebraic: $\$\mathbf{0} = \mathbf{f}(\mathbf{x}, \mathbf{y})\mathbf{0} = \mathbf{g}(\mathbf{x}, \mathbf{y})\$$

Solved using Newton-Raphson iteration.

Time-Domain Simulation

Differential equations are integrated over time using implicit methods (trapezoidal rule).

At each time step, discretized equations become: $\mathbf{x}_{n+1} - \mathbf{x}_n = \frac{h}{2}(\mathbf{f}_{n+1} + \mathbf{f}_n)$

Combined with algebraic constraints, solved iteratively.

Jacobian Matrices

For Newton-Raphson iteration, we need:

$$\mathbf{J} = \begin{bmatrix} \frac{\partial \mathbf{f}}{\partial \mathbf{x}} & \frac{\partial \mathbf{f}}{\partial \mathbf{y}} \\ \frac{\partial \mathbf{g}}{\partial \mathbf{x}} & \frac{\partial \mathbf{g}}{\partial \mathbf{y}} \end{bmatrix}$$

ANDES computes these automatically from symbolic equations:

```
ss.dae.fx     # df/dx
ss.dae.fy     # df/dy
ss.dae.gx     # dg/dx
ss.dae.gy     # dg/dy
```

Equation Conventions

Differential Equations

Written in the form: $M\dot{x} = f(x, y)$

```
# Swing equation: M * d(omega)/dt = Tm - Te - D*(omega-1)
self.omega = State(e_str='Tm - Te - D*(omega - 1)',
                   t_const=self.M)
```

The time constant `t_const` specifies the coefficient on the left-hand side of the differential equation (the mass matrix diagonal entry). It is not included in `e_str`, which only defines the right-hand side $f(x, y)$.

Algebraic Equations

Written in the form: $0 = g(x, y)$

```
# Power balance: 0 = P_gen - P_load - P_flow
self.P = Algeb(e_str='Pg - Pl - Pflow')
```

Initialization

Before simulation, initial values must satisfy:

$$\mathbf{0} = \mathbf{f}(\mathbf{x}_0, \mathbf{y}_0)$$

$$\mathbf{0} = \mathbf{g}(\mathbf{x}_0, \mathbf{y}_0)$$

ANDES supports two initialization methods:

Explicit (`v_str`)

Direct calculation of initial value:

```
self.omega = State(v_str='1.0')  # omega starts at 1.0 pu
```

Implicit (`v_iter`)

Solve iteratively using Newton-Krylov:

```
self.Efd = Algeb(v_iter='Vf - Efd')  # Solve Efd such that Vf - Efd = 0
```

Handling Discontinuities

Power systems have discontinuous behavior:

- Limits (saturation, rate limits)
- Switching (breaker operations)
- Dead bands

ANDES uses **discrete components** that export flags:

```
self.lim = Limiter(u=self.x, lower=-1, upper=1)
# Exports: lim.zi (within), lim.zl (below), lim.zu (above)

# Use in equation
self.y = Algeb(e_str='x * lim_zi + upper * lim_zu + lower * lim_zl')
```

Mass Matrix

The mass matrix $\mathbf{M}$ is diagonal and stored in `dae.Tf`. Each `State` variable's `t_const` value populates the corresponding diagonal entry. For states without `t_const`, the entry defaults to 1 (identity).

Model developers do not need to manipulate the mass matrix directly. Simply specify `t_const` when defining state variables:

```
# t_const value goes to dae.Tf[omega.a], where omega.a is the address
self.omega = State(e_str='Tm - Te - D*(omega - 1)',
                   t_const=self.M)
```

This automatically handles the left-hand side coefficient, keeping `e_str` focused on the physics.

Eigenvalue Analysis

At equilibrium, linearize around $(\mathbf{x}_0, \mathbf{y}_0)$:

$$\Delta\dot{\mathbf{x}} = \mathbf{A}_s \Delta\mathbf{x}$$

Where the state matrix is:

$$\mathbf{A}_s = \mathbf{f}_x - \mathbf{f}_y \mathbf{g}_y^{-1} \mathbf{g}_x$$

Eigenvalues of $\mathbf{A}_s$ determine stability.

See Also

- *Atomic Types* - v-provider and e-provider concepts
- *Hybrid Symbolic-Numeric Framework* - Symbolic framework overview
- *Variables* - Variable types in detail
- *Discrete Components* - Discrete components

4.3 Model Components

Building blocks for defining power system models.

See also:

Before diving into specific components, understand the foundational concepts:

- *Atomic Types* - How parameters, variables, and services share a common interface (v-provider)
- *DAE Formulation* - How variables contribute to the DAE system (e-provider)

4.3.1 Parameters

Parameters are a type of building atom for DAE models. Most parameters are read directly from an input file and passed to equations, though some can be calculated from existing parameters.

Background

The base class for parameters in ANDES is `BaseParam`, which defines interfaces for adding values and checking the number of values. `BaseParam` stores values in a plain list via the member attribute `v`. Subclasses such as `NumParam` store values using a NumPy ndarray for efficient numerical computation.

All parameters are **v-providers**: they have a `v` attribute containing values that can be used in equation strings. This shared interface enables parameters and services to be used interchangeably in equations. See *Atomic Types* for details on the v-provider concept.

Parameter Types

BaseParam([default, name, tex_name, info, ...])	The base parameter class.
NumParam(default, name, tex_name, info, ...)	A computational numerical parameter.
IdxParam([default, name, tex_name, info, ...])	Parameter for storing idx references into other models.
ExtParam(model, src[, indexer, vtype, ...])	A parameter whose values are retrieved from an external model or group.
DataParam([default, name, tex_name, info, ...])	An alias of the *BaseParam* class.
TimerParam([callback, default, name, ...])	A parameter whose values are event occurrence times during the simulation.

andes.core.param.BaseParam

class andes.core.param.**BaseParam**(*default: float | str | int | None = None, name: str | None = None, tex_name: str | None = None, info: str | None = None, unit: str | None = None, mandatory: bool = False, export: bool = True, iconvert: Callable | None = None, oconvert: Callable | None = None*)

The base parameter class.

This class provides the basic data structure and interfaces for all types of parameters. Parameters are from input files and in general constant once initialized.

Subclasses should overload the *n()* method for the total count of elements in the value array.

Parameters

default
[str or float, optional] The default value of this parameter if None is provided

name
[str, optional] Parameter name. If not provided, it will be automatically set to the attribute name defined in the owner model.

tex_name
[str, optional] LaTeX-formatted parameter name. If not provided, *tex_name* will be assigned the same as *name*.

info
[str, optional] Descriptive information of parameter

mandatory
[bool] True if this parameter is mandatory

export
[bool] True if the parameter will be exported when dumping data into files. True for most parameters. False for `BackRef`.

Attributes

v
[list] A list holding all the values. The `BaseParam` class does not convert the `v` attribute into NumPy arrays.

property
[dict] A dict containing the truth values of the model properties.

Other Parameters

iconvert
[Callable] Converter to be applied to input data when a device is being added.

oconvert
[callable] Converter to be applied to internal data when outputting.

> **Warning:** The most distinct feature of BaseParam, DataParam and IdxParam is that values are stored in a list without conversion to array. BaseParam, DataParam or IdxParam are **not allowed** in equations.

`__init__`(*default: float | str | int | None = None, name: str | None = None, tex_name: str | None = None, info: str | None = None, unit: str | None = None, mandatory: bool = False, export: bool = True, iconvert: Callable | None = None, oconvert: Callable | None = None*)

Methods

`add`([value])	Add a new parameter value (from a new device of the owner model) to the `v` list.
`get_names`()	Return `self.name` in a list.
`get_property`(property_name)	Check the boolean value of the given property.
`set`(pos, attr, value)	Set attributes of the BaseParam class to new values at the given positions.
`set_all`(attr, value)	Set attributes of the BaseParam class to new values for all positions.

Attributes

`class_name`	Return the class name.
`n`	Return the count of elements in the value array.

andes.core.param.NumParam

class andes.core.param.**NumParam**(*default: float | str | ~typing.Callable | None = None, name: str | None = None, tex_name: str | None = None, info: str | None = None, unit: str | None = None, vrange: ~typing.List | ~typing.Tuple | None = None, vtype: ~typing.Type | None = <class 'float'>, iconvert: ~typing.Callable | None = None, oconvert: ~typing.Callable | None = None, non_zero: bool = False, non_positive: bool = False, non_negative: bool = False, mandatory: bool = False, power: bool = False, ipower: bool = False, voltage: bool = False, current: bool = False, z: bool = False, y: bool = False, r: bool = False, g: bool = False, dc_voltage: bool = False, dc_current: bool = False, export: bool = True*)

A computational numerical parameter.

Parameters defined using this class will have their *v* field converted to a NumPy array after adding.

The original input values will be copied to *vin*, and the system-base per-unit conversion coefficients (through multiplication) will be stored in *pu_coeff*.

Parameters

default
[str or float, optional] The default value of this parameter if no value is provided

name
[str, optional] Name of this parameter. If not provided, *name* will be set to the attribute name of the owner model.

tex_name
[str, optional] LaTeX-formatted parameter name. If not provided, *tex_name* will be assigned the same as *name*.

info
[str, optional] A description of this parameter

mandatory
[bool] True if this parameter is mandatory

unit
[str, optional] Unit of the parameter

vrange
[list, tuple, optional] Typical value range

vtype
[type, optional] Type of the `v` field. The default is `float`.

Attributes

vin
[np.ndarray or None] Raw input values before per-unit conversion. Used by `restore()` (pre-pu restore for `System.reset`).

_v_t0
[np.ndarray or None] Post-pu `v` at t=0, saved by `snapshot_init()`. Used by `restore_init()` to reset `v` during `TDS.reinit`.

Other Parameters

Sn
[str] Name of the parameter for the device base power.

Vn
[str] Name of the parameter for the device base voltage.

non_zero
[bool] True if this parameter must be non-zero. *non_zero* can be combined with *non_positive* or *non_negative*.

non_positive
[bool] True if this parameter must be non-positive.

non_negative
[bool] True if this parameter must be non-negative.

mandatory
[bool] True if this parameter must not be None.

power
[bool] True if this parameter is a power per-unit quantity under the device base.

iconvert
[callable] Callable to convert input data from excel or others to the internal `v` field.

oconvert
[callable] Callable to convert input data from internal type to a serializable type.

ipower
[bool] True if this parameter is an inverse-power per-unit quantity under the device base.

voltage
[bool] True if the parameter is a voltage pu quantity under the device base.

current
[bool] True if the parameter is a current pu quantity under the device base.

z
[bool] True if the parameter is an AC impedance pu quantity under the device base.

y
[bool] True if the parameter is an AC admittance pu quantity under the device base.

r
[bool] True if the parameter is a DC resistance pu quantity under the device base.

g
[bool] True if the parameter is a DC conductance pu quantity under the device base.

dc_current
[bool] True if the parameter is a DC current pu quantity under device base.

dc_voltage
[bool] True if the parameter is a DC voltage pu quantity under device base.

__init__(*default: float | str | ~typing.Callable | None = None*, *name: str | None = None*, *tex_name: str | None = None*, *info: str | None = None*, *unit: str | None = None*, *vrange: ~typing.List | ~typing.Tuple | None = None*, *vtype: ~typing.Type | None = <class 'float'>*, *iconvert: ~typing.Callable | None = None*, *oconvert: ~typing.Callable | None = None*, *non_zero: bool = False*, *non_positive: bool = False*, *non_negative: bool = False*, *mandatory: bool = False*, *power: bool = False*, *ipower: bool = False*, *voltage: bool = False*, *current: bool = False*, *z: bool = False*, *y: bool = False*, *r: bool = False*, *g: bool = False*, *dc_voltage: bool = False*, *dc_current: bool = False*, *export: bool = True*)

Methods

add([value])	Add a value to the parameter value list.
get_names()	Return self.name in a list.
get_property(property_name)	Check the boolean value of the given property.
restore()	Restore parameter to the original input by copying self.vin to self.v.
restore_init()	Restore v from _v_t0 saved by snapshot_init().
set(pos, attr, value)	Set attributes of the BaseParam class to new values at the given positions.
set_all(attr, value)	Set attributes of the BaseParam class to new values for all positions.
set_pu_coeff(coeff)	Store p.u.
snapshot_init()	Save the current post-pu v into _v_t0 for restore_init().
to_array()	Converts field v to the NumPy array type.

Attributes

class_name	Return the class name.
n	Return the count of elements in the value array.

andes.core.param.IdxParam

class andes.core.param.**IdxParam**(*default: float | str | int | None = None, name: str | None = None, tex_name: str | None = None, info: str | None = None, unit: str | None = None, mandatory: bool = False, unique: bool = False, export: bool = True, model: str | None = None, iconvert: Callable | None = None, oconvert: Callable | None = None, replaces: bool = False, status_parent: bool = False*)

Parameter for storing idx references into other models.

IdxParam creates the connection graph between models. It stores device idx values that point to entries in the referenced model (which can be a Model or Group name). These references are used by ExtParam and ExtAlgeb/ExtState to look up external values.

Parameters

model

[str, optional] Name of the referenced Model or Group (e.g. 'Bus', 'StaticGen'). Required for BackRef and external-variable linking.

unique
[bool, optional] If `True`, duplicate values raise `IndexError`. Useful when a one-to-one mapping is required (e.g. one TG per generator).

replaces
[bool, optional] If `True`, this model replaces the referenced device during time-domain simulation (static-dynamic replacement).

status_parent
[bool, optional] If `True`, the referenced device is treated as a status parent. When the parent is taken offline via `set_status`, the effective status `ue` of this device and its descendants is propagated automatically.

Examples

A PQ model connected to Bus

```
class PQModel(...):
    def __init__(...):
        self.bus = IdxParam(model='Bus')
```

An exciter referencing its parent generator with status propagation

```
self.syn = IdxParam(model='SynGen', mandatory=True,
                    status_parent=True)
```

__init__(*default: float | str | int | None = None, name: str | None = None, tex_name: str | None = None, info: str | None = None, unit: str | None = None, mandatory: bool = False, unique: bool = False, export: bool = True, model: str | None = None, iconvert: Callable | None = None, oconvert: Callable | None = None, replaces: bool = False, status_parent: bool = False*)

Methods

`add`([value])	Add a new parameter value (from a new device of the owner model) to the `v` list.
`get_names`()	Return `self.name` in a list.
`get_property`(property_name)	Check the boolean value of the given property.
`set`(pos, attr, value)	Set attributes of the BaseParam class to new values at the given positions.
`set_all`(attr, value)	Set attributes of the BaseParam class to new values for all positions.

Attributes

`class_name`	Return the class name.
`n`	Return the count of elements in the value array.

andes.core.param.ExtParam

class `andes.core.param.`**`ExtParam`**(*model: str*, *src: str*, *indexer=None*, *vtype=<class 'float'>*, *allow_none=False*, *default=0.0*, ***kwargs*)

A parameter whose values are retrieved from an external model or group.

Parameters

model
[str] Name of the model or group providing the original parameter

src
[str] The source parameter name

indexer
[BaseParam] A parameter defined in the model defining this ExtParam instance. *indexer.v* should contain indices into *model.src.v*. If is None, the source parameter values will be fully copied. If *model* is a group name, the indexer cannot be None.

vtype
[type, optional, default to float] Type of each element to be retrieved. Can be `str` if the ExtParam is used to access an `IdxParam`.

Attributes

parent_model
[Model] The parent model providing the original parameter.

`__init__`(*model: str*, *src: str*, *indexer=None*, *vtype=<class 'float'>*, *allow_none=False*, *default=0.0*, ***kwargs*)

Methods

add([value])	ExtParam has an empty *add* method.
get_names()	Return `self.name` in a list.
get_property(property_name)	Check the boolean value of the given property.
link_external(ext_model)	Update parameter values provided by external models.
restore()	ExtParam has an empty *restore* method
restore_init()	Restore `v` from `_v_t0` saved by `snapshot_init()`.
set(pos, attr, value)	Set attributes of the BaseParam class to new values at the given positions.
set_all(attr, value)	Set attributes of the BaseParam class to new values for all positions.
set_pu_coeff(coeff)	Store p.u.
snapshot_init()	Save the current post-pu `v` into `_v_t0` for `restore_init()`.
to_array()	Convert to array when d_type is not str

Attributes

class_name	Return the class name.
n	Return the count of elements in the value array.

andes.core.param.DataParam

class andes.core.param.**DataParam**(*default: float | str | int | None = None, name: str | None = None, tex_name: str | None = None, info: str | None = None, unit: str | None = None, mandatory: bool = False, export: bool = True, iconvert: Callable | None = None, oconvert: Callable | None = None*)

An alias of the *BaseParam* class.

This class is used for string parameters or non-computational numerical parameters. This class does not provide a *to_array* method. All input values will be stored in *v* as a list.

See also:

andes.core.param.BaseParam
Base parameter class

__init__(*default: float | str | int | None = None, name: str | None = None, tex_name: str | None = None, info: str | None = None, unit: str | None = None, mandatory: bool = False, export: bool = True, iconvert: Callable | None = None, oconvert: Callable | None = None*)

Methods

add([value])	Add a new parameter value (from a new device of the owner model) to the v list.
get_names()	Return self.name in a list.
get_property(property_name)	Check the boolean value of the given property.
set(pos, attr, value)	Set attributes of the BaseParam class to new values at the given positions.
set_all(attr, value)	Set attributes of the BaseParam class to new values for all positions.

Attributes

class_name	Return the class name.
n	Return the count of elements in the value array.

andes.core.param.TimerParam

class andes.core.param.**TimerParam**(*callback: Callable | None = None, default: float | str | Callable | None = None, name: str | None = None, tex_name: str | None = None, info: str | None = None, unit: str | None = None, non_zero: bool = False, mandatory: bool = False, export: bool = True*)

A parameter whose values are event occurrence times during the simulation.

The constructor takes an additional Callable *self.callback* for the action of the event. *TimerParam* has a default value of -1, meaning deactivated.

Examples

A connectivity status toggle class *Toggle* takes a parameter *t* for the toggle time. Inside Toggle.__init__, one would have

```
self.t = TimerParam()
```

The *Toggle* class also needs to define a method for togging the connectivity status

```
def _u_switch(self, is_time: np.ndarray):
    action = False
    for i in range(self.n):
```

(continues on next page)

(continued from previous page)

```
        if is_time[i] and (self.u.v[i] == 1):
            instance = self.system.__dict__[self.model.v[i]]
            # get the original status and flip the value
            u0 = instance.get(src='u', attr='v', idx=self.dev.v[i])
            instance.set(src='u',
                         attr='v',
                         idx=self.dev.v[i],
                         value=1-u0)
            action = True
    return action
```

Finally, in `Toggle.__init__`, assign the function as the callback for *self.t*

```
self.t.callback = self._u_switch
```

__init__(*callback: Callable | None = None, default: float | str | Callable | None = None, name: str | None = None, tex_name: str | None = None, info: str | None = None, unit: str | None = None, non_zero: bool = False, mandatory: bool = False, export: bool = True*)

Methods

`add`([value])	Add a value to the parameter value list.
`get_names`()	Return `self.name` in a list.
`get_property`(property_name)	Check the boolean value of the given property.
`is_time`(dae_t)	Element-wise check if the DAE time is the same as the parameter value.
`restore`()	Restore parameter to the original input by copying `self.vin` to `self.v`.
`restore_init`()	Restore `v` from `_v_t0` saved by `snapshot_init()`.
`set`(pos, attr, value)	Set attributes of the BaseParam class to new values at the given positions.
`set_all`(attr, value)	Set attributes of the BaseParam class to new values for all positions.
`set_pu_coeff`(coeff)	Store p.u.
`snapshot_init`()	Save the current post-pu `v` into `_v_t0` for `restore_init()`.
`to_array`()	Converts field `v` to the NumPy array type.

Attributes

class_name	Return the class name.
n	Return the count of elements in the value array.

NumParam

The most common parameter type for numerical values that participate in equations.

```
from andes.core.param import NumParam

class MyModel(Model):
    def __init__(self):
        # Basic parameter
        self.R = NumParam(default=0.05,
                          info='Droop coefficient',
                          tex_name='R')

        # Parameter with constraints
        self.H = NumParam(default=3.0,
                          info='Inertia constant',
                          non_zero=True,
                          non_negative=True)

        # Parameter with units
        self.Vn = NumParam(default=20.0,
                           info='Rated voltage',
                           unit='kV')
```

`NumParam` values are stored in a NumPy array accessible via `self.R.v`. When used in equation strings like `e_str='1/R'`, the parameter name is automatically substituted with its values during evaluation.

class andes.core.param.**NumParam**(*default: float | str | ~typing.Callable | None = None, name: str | None = None, tex_name: str | None = None, info: str | None = None, unit: str | None = None, vrange: ~typing.List | ~typing.Tuple | None = None, vtype: ~typing.Type | None = <class 'float'>, iconvert: ~typing.Callable | None = None, oconvert: ~typing.Callable | None = None, non_zero: bool = False, non_positive: bool = False, non_negative: bool = False, mandatory: bool = False, power: bool = False, ipower: bool = False, voltage: bool = False, current: bool = False, z: bool = False, y: bool = False, r: bool = False, g: bool = False, dc_voltage: bool = False, dc_current: bool = False, export: bool = True*)

A computational numerical parameter.

Parameters defined using this class will have their *v* field converted to a NumPy array after adding.

The original input values will be copied to *vin*, and the system-base per-unit conversion coefficients (through multiplication) will be stored in *pu_coeff*.

Parameters

default
[str or float, optional] The default value of this parameter if no value is provided

name
[str, optional] Name of this parameter. If not provided, *name* will be set to the attribute name of the owner model.

tex_name
[str, optional] LaTeX-formatted parameter name. If not provided, *tex_name* will be assigned the same as *name*.

info
[str, optional] A description of this parameter

mandatory
[bool] True if this parameter is mandatory

unit
[str, optional] Unit of the parameter

vrange
[list, tuple, optional] Typical value range

vtype
[type, optional] Type of the `v` field. The default is `float`.

Attributes

vin
[np.ndarray or None] Raw input values before per-unit conversion. Used by `restore()` (pre-pu restore for `System.reset`).

_v_t0
[np.ndarray or None] Post-pu `v` at t=0, saved by `snapshot_init()`. Used by `restore_init()` to reset `v` during `TDS.reinit`.

Other Parameters

Sn
[str] Name of the parameter for the device base power.

Vn
[str] Name of the parameter for the device base voltage.

non_zero
[bool] True if this parameter must be non-zero. *non_zero* can be combined with *non_positive* or *non_negative*.

non_positive
[bool] True if this parameter must be non-positive.

non_negative
[bool] True if this parameter must be non-negative.

mandatory
[bool] True if this parameter must not be None.

power
[bool] True if this parameter is a power per-unit quantity under the device base.

iconvert
[callable] Callable to convert input data from excel or others to the internal `v` field.

oconvert
[callable] Callable to convert input data from internal type to a serializable type.

ipower
[bool] True if this parameter is an inverse-power per-unit quantity under the device base.

voltage
[bool] True if the parameter is a voltage pu quantity under the device base.

current
[bool] True if the parameter is a current pu quantity under the device base.

z
[bool] True if the parameter is an AC impedance pu quantity under the device base.

y
[bool] True if the parameter is an AC admittance pu quantity under the device base.

r
[bool] True if the parameter is a DC resistance pu quantity under the device base.

g
[bool] True if the parameter is a DC conductance pu quantity under the device base.

dc_current
[bool] True if the parameter is a DC current pu quantity under device base.

dc_voltage
[bool] True if the parameter is a DC voltage pu quantity under device base.

add(*value=None*)

Add a value to the parameter value list.

In addition to `BaseParam.add`, this method checks for non-zero property and reset to default if is zero.

Returns

str or None
The violation type string ('non_zero', 'non_positive', 'non_negative') if a correction was applied, or None if no correction was needed.

See also:

BaseParam.add
the add method of BaseParam

restore()

Restore parameter to the original input by copying `self.vin` to `self.v`.

pu_coeff will not be overwritten.

restore_init()

Restore `v` from `_v_t0` saved by `snapshot_init()`.

This restores post-pu values directly. Contrast with `restore()`, which copies pre-pu `vin` into `v`.

Called by `Model.restore_init` during `TDS.reinit()`.

set_pu_coeff(*coeff*)

Store p.u. conversion coefficient into `self.pu_coeff` and calculate the system-base per unit with `self.v = self.vin * self.pu_coeff`.

This function must be called after `self.to_array`.

Parameters

coeff
[np.ndarray] An array with the pu conversion coefficients

snapshot_init()

Save the current post-pu `v` into `_v_t0` for `restore_init()`.

Unlike `restore()` which copies pre-pu `vin` back to `v`, this saves the fully converted (post-pu) values so that `restore_init` can reset `v` without re-running pu conversion.

Called by `Model.snapshot_init` at the end of `TDS.init()`.

to_array()

Converts field `v` to the NumPy array type. to enable array-based calculation.

Must be called after adding all elements. Store a copy of original input values to field `vin`. Set `pu_coeff` to all ones.

> **Warning:** After this call, *add* will not be allowed to avoid unexpected issues.

IdxParam

References to devices in other models. `IdxParam` creates the connection graph between models, enabling external parameters and variables to look up values from referenced devices.

```
from andes.core.param import IdxParam

class PQ(Model):
```

(continues on next page)

(continued from previous page)

```
    def __init__(self):
        # Reference to Bus model
        self.bus = IdxParam(model='Bus',
                            mandatory=True,
                            info='Connected bus idx')
```

The connection graph enables model relationships:

```
GENROU.bus → Bus.idx
GENROU.gen → StaticGen.idx
ESST1A.syn → GENROU.idx
```

Set `status_parent=True` to declare that the referenced device is a status parent. When a parent device is taken offline via `set_status`, the effective status (`ue`) of the child and its descendants is automatically propagated:

```
self.syn = IdxParam(model='SynGen', mandatory=True, status_parent=True)
```

class andes.core.param.**IdxParam**(*default: float | str | int | None = None, name: str | None = None, tex_name: str | None = None, info: str | None = None, unit: str | None = None, mandatory: bool = False, unique: bool = False, export: bool = True, model: str | None = None, iconvert: Callable | None = None, oconvert: Callable | None = None, replaces: bool = False, status_parent: bool = False*)

Parameter for storing idx references into other models.

`IdxParam` creates the connection graph between models. It stores device idx values that point to entries in the referenced `model` (which can be a Model or Group name). These references are used by `ExtParam` and `ExtAlgeb`/`ExtState` to look up external values.

Parameters

model
[str, optional] Name of the referenced Model or Group (e.g. `'Bus'`, `'StaticGen'`). Required for BackRef and external-variable linking.

unique
[bool, optional] If `True`, duplicate values raise `IndexError`. Useful when a one-to-one mapping is required (e.g. one TG per generator).

replaces
[bool, optional] If `True`, this model replaces the referenced device during time-domain simulation (static-dynamic replacement).

status_parent
[bool, optional] If `True`, the referenced device is treated as a status parent. When the parent is taken offline via `set_status`, the effective status `ue` of this device and its descendants is propagated automatically.

Examples

A PQ model connected to Bus

```
class PQModel(...):
    def __init__(...):
        self.bus = IdxParam(model='Bus')
```

An exciter referencing its parent generator with status propagation

```
self.syn = IdxParam(model='SynGen', mandatory=True,
                    status_parent=True)
```

add(*value=None*)

Add a new parameter value (from a new device of the owner model) to the `v` list.

Parameters

value
[str or float, optional] Parameter value of the new element. If None, the default will be used.

Notes

If the value is `math.nan`, it will set to `None`.

ExtParam

Retrieves parameter values from external models. Uses an `IdxParam` as indexer to look up values from the referenced model.

```
from andes.core.param import ExtParam

class GENROU(Model):
    def __init__(self):
        # Get bus voltage from connected bus
        self.Vn = ExtParam(src='Vn',
                           model='Bus',
                           indexer=self.bus)

        # Get initial power from static generator
        self.p0 = ExtParam(src='p',
                           model='StaticGen',
                           indexer=self.gen)
```

`ExtParam` is essential for models that need configuration or initial conditions from other models. The retrieved values are stored locally in a NumPy array and can be used in equation strings just like `NumParam`.

class andes.core.param.**ExtParam**(*model: str, src: str, indexer=None, vtype=<class 'float'>, allow_none=False, default=0.0, **kwargs*)

A parameter whose values are retrieved from an external model or group.

Parameters

model
[str] Name of the model or group providing the original parameter

src
[str] The source parameter name

indexer
[BaseParam] A parameter defined in the model defining this ExtParam instance. *indexer.v* should contain indices into *model.src.v*. If is None, the source parameter values will be fully copied. If *model* is a group name, the indexer cannot be None.

vtype
[type, optional, default to float] Type of each element to be retrieved. Can be `str` if the ExtParam is used to access an `IdxParam`.

Attributes

parent_model
[Model] The parent model providing the original parameter.

add(*value=None*)

ExtParam has an empty *add* method.

link_external(*ext_model*)

Update parameter values provided by external models. This needs to be called before pu conversion.

Parameters

ext_model
[Model, Group] Instance of the parent model or group, provided by the System calling this method.

restore()

ExtParam has an empty *restore* method

to_array()

Convert to array when d_type is not str

DataParam

For non-numeric data such as names or string identifiers that should not participate in numerical calculations.

```
from andes.core.param import DataParam

class Bus(Model):
    def __init__(self):
        self.name = DataParam(info='Bus name')
```

class andes.core.param.**DataParam**(*default: float | str | int | None = None, name: str | None = None, tex_name: str | None = None, info: str | None = None, unit: str | None = None, mandatory: bool = False, export: bool = True, iconvert: Callable | None = None, oconvert: Callable | None = None*)

An alias of the *BaseParam* class.

This class is used for string parameters or non-computational numerical parameters. This class does not provide a *to_array* method. All input values will be stored in *v* as a list.

See also:

andes.core.param.BaseParam
Base parameter class

TimerParam

For time-based events. `TimerParam` stores callback functions that are triggered when simulation time reaches the parameter value. This parameter type enables the timed event mechanism used by `Fault`, `Toggle`, and `Alter` models.

```
from andes.core.param import TimerParam

class Fault(Model):
    def __init__(self):
        self.tf = TimerParam(info='Fault start time',
                             callback=self.apply_fault)
        self.tc = TimerParam(info='Fault clear time',
                             callback=self.clear_fault)
```

The callback function receives a boolean array indicating which devices have reached their trigger time. See *System Architecture* for details on how the timed event mechanism works internally.

class andes.core.param.**TimerParam**(*callback: Callable | None = None, default: float | str | Callable | None = None, name: str | None = None, tex_name: str | None = None, info: str | None = None, unit: str | None = None, non_zero: bool = False, mandatory: bool = False, export: bool = True*)

A parameter whose values are event occurrence times during the simulation.

The constructor takes an additional Callable *self.callback* for the action of the event. *TimerParam* has a default value of -1, meaning deactivated.

Examples

A connectivity status toggle class *Toggle* takes a parameter *t* for the toggle time. Inside `Toggle.__init__`, one would have

```
self.t = TimerParam()
```

The *Toggle* class also needs to define a method for togging the connectivity status

```
def _u_switch(self, is_time: np.ndarray):
    action = False
    for i in range(self.n):
        if is_time[i] and (self.u.v[i] == 1):
            instance = self.system.__dict__[self.model.v[i]]
            # get the original status and flip the value
            u0 = instance.get(src='u', attr='v', idx=self.dev.v[i])
            instance.set(src='u',
                         attr='v',
                         idx=self.dev.v[i],
                         value=1-u0)
            action = True
    return action
```

Finally, in `Toggle.__init__`, assign the function as the callback for *self.t*

```
self.t.callback = self._u_switch
```

is_time(*dae_t*)

Element-wise check if the DAE time is the same as the parameter value. The current implementation uses *np.equal*.

Parameters

dae_t
[float] Current simulation time

Returns

np.ndarray
The array containing the truth value of if the DAE time is close to the parameter value.

Notes

The previous implementation with *np.isclose* with default *rtol=1e-5* mistakes the immediate pre- and post-event time as in-event when simulation time is greater than 10.

BaseParam

The abstract base class defining the parameter interface. All parameter types inherit from `BaseParam`.

class `andes.core.param.`**`BaseParam`**(*default: float | str | int | None = None, name: str | None = None, tex_name: str | None = None, info: str | None = None, unit: str | None = None, mandatory: bool = False, export: bool = True, iconvert: Callable | None = None, oconvert: Callable | None = None*)

The base parameter class.

This class provides the basic data structure and interfaces for all types of parameters. Parameters are from input files and in general constant once initialized.

Subclasses should overload the *n()* method for the total count of elements in the value array.

Parameters

default
[str or float, optional] The default value of this parameter if None is provided

name
[str, optional] Parameter name. If not provided, it will be automatically set to the attribute name defined in the owner model.

tex_name
[str, optional] LaTeX-formatted parameter name. If not provided, *tex_name* will be assigned the same as *name*.

info
[str, optional] Descriptive information of parameter

mandatory
[bool] True if this parameter is mandatory

export
[bool] True if the parameter will be exported when dumping data into files. True for most parameters. False for `BackRef`.

Attributes

v
[list] A list holding all the values. The `BaseParam` class does not convert the `v` attribute into NumPy arrays.

property
[dict] A dict containing the truth values of the model properties.

Other Parameters

iconvert
[Callable] Converter to be applied to input data when a device is being added.

oconvert
[callable] Converter to be applied to internal data when outputting.

> **Warning:** The most distinct feature of BaseParam, DataParam and IdxParam is that values are stored in a list without conversion to array. BaseParam, DataParam or IdxParam are **not allowed** in equations.

add(*value=None*)

Add a new parameter value (from a new device of the owner model) to the `v` list.

Parameters

value
[str or float, optional] Parameter value of the new element. If None, the default will be used.

Notes

If the value is `math.nan`, it will set to `None`.

property class_name

Return the class name.

get_names()

Return `self.name` in a list.

This is a helper function to provide the same API as blocks or discrete components.

Returns

list
A list only containing the name of the parameter

get_property(*property_name: str*)

Check the boolean value of the given property. If the property does not exist in the dictionary, `False` will be returned.

Parameters

property_name
[str] Property name

Returns

The truth value of the property.

property n

Return the count of elements in the value array.

set(*pos*, *attr*, *value*)

Set attributes of the BaseParam class to new values at the given positions.

Parameters

pos
[int, list of integers] Positions in arrays where the values should be set

attr
['v', 'vin'] Name of the attribute to be set

value
[str, float or list of above] New values

set_all (*attr*, *value*)

Set attributes of the BaseParam class to new values for all positions.

Parameters

attr
['v', 'vin'] Name of the attribute to be set

value
[list of str, float or int] New values

Parameters in Equations

Parameters can be used directly in equation strings by name. The symbolic framework substitutes parameter names with their `v` arrays during evaluation:

```
self.R = NumParam(default=0.05)
self.D = NumParam(default=2.0)

# Use parameters in service calculation
self.gain = ConstService(v_str='1/R')

# Use parameters in differential equation
self.omega = State(e_str='Tm - Pe - D*(omega-1)')
```

Mandatory vs Optional

- **Mandatory**: Must be provided in input data; simulation fails without it
- **Optional**: Uses default value if not provided

```
self.bus = IdxParam(mandatory=True)    # Required
self.D = NumParam(default=0.0)         # Optional, defaults to 0
```

Input Validation

ANDES validates parameters after loading:

- `NaN` values raise `ValueError`
- `inf` values are replaced with `1e8`
- Constraint violations (e.g., `non_zero`) use defaults with a warning

See Also

- *Atomic Types* - v-provider and e-provider concepts
- *Variables* - Variable types that use parameters in equations
- *Services* - Service calculations using parameters

4.3.2 Variables

DAE Variables, or variables for short, are unknowns to be solved using numerical or analytical methods. A variable stores values, equation values, and addresses in the DAE array.

Background

Variables are both **v-providers** and **e-providers** (see *Atomic Types*). Each variable has member attributes `v` and `e` for variable values and equation values, respectively. The initial value of `v` is set by the initialization routine, and the initial value of `e` is set to zero.

During power flow calculation or time domain simulation:

- `v` is not directly modifiable by models but rather updated after solving non-linear equations
- `e` is updated by the models and summed up before solving equations

Each variable also stores addresses in its member attribute `a`. The addresses are *0-based* indices into the numerical DAE array (`f` or `g`, based on variable type).

Equations associated with state variables take the form:

$$\mathbf{M}\dot{x} = \mathbf{f}(x, y)$$

where **x** are the differential variables, **y** are the algebraic variables, **M** is the mass matrix, and **f** is the right-hand side of differential equations.

Equations associated with algebraic variables take the form:

$$0 = \mathbf{g}(x, y)$$

where **g** is the equation right-hand side.

Variable Types

`BaseVar`([name, tex_name, info, unit, v_str, ...])	Base variable class.
`State`([name, tex_name, info, unit, v_str, ...])	Differential variable class, an alias of the *BaseVar*.
`Algeb`([name, tex_name, info, unit, v_str, ...])	Algebraic variable class, an alias of `andes.core.var.BaseVar`.
`ExtVar`(model, src[, indexer, allow_none, ...])	Algebraic variable that links to an external model.
`ExtState`(model, src[, indexer, allow_none, ...])	External state variable type.
`ExtAlgeb`(model, src[, indexer, allow_none, ...])	External algebraic variable type.
`AliasState`(var, **kwargs)	Alias state variable.
`AliasAlgeb`(var, **kwargs)	Alias algebraic variable.

andes.core.var.BaseVar

class andes.core.var.**BaseVar**(*name: str | None = None, tex_name: str | None = None, info: str | None = None, unit: str | None = None, v_str: str | float | None = None, v_iter: str | None = None, e_str: str | None = None, discrete: Discrete | None = None, v_setter: bool | None = False, e_setter: bool | None = False, v_str_add: bool | None = False, addressable: bool | None = True, export: bool | None = True, diag_eps: float | None = 0.0, deps: List | None = None, is_output: bool | None = False*)

Base variable class.

Derived classes *State* and *Algeb* should be used to build model variables.

Parameters

info
[str, optional] Descriptive information

unit
[str, optional] Unit

tex_name
[str] LaTeX-formatted variable symbol. If is None, the value of *name* will be used.

discrete
[Discrete] Discrete component on which this variable depends. ANDES will call *check_var()* of the discrete component before initializing this variable.

name
[str, optional] Variable name. One should typically assigning the name directly because it will be automatically assigned by the model. The value of `name` will be the symbol name to be used in expressions.

Attributes

a
[array-like] variable address

v
[array-like] local-storage of the variable value

e
[array-like] local-storage of the corresponding equation value

e_str
[str] the string/symbolic representation of the equation

v_str
[str] explicit initialization equation

v_str_add
[bool] True if the value of *v_str* will be added to the variable. Useful when other models access this variable and set part of the initial value

v_iter
[str] implicit iterative equation in the form of 0 = v_iter

__init__(*name: str | None = None, tex_name: str | None = None, info: str | None = None, unit: str | None = None, v_str: str | float | None = None, v_iter: str | None = None, e_str: str | None = None, discrete: Discrete | None = None, v_setter: bool | None = False, e_setter: bool | None = False, v_str_add: bool | None = False, addressable: bool | None = True, export: bool | None = True, diag_eps: float | None = 0.0, deps: List | None = None, is_output: bool | None = False*)

Methods

`get_names()`	
`reset()`	Reset the internal numpy arrays and flags.
`set_address`(addr[, contiguous])	Set the address of internal variables.
`set_arrays`(dae[, inplace, alloc])	Set the equation and values arrays.

Attributes

`class_name`

andes.core.var.State

class andes.core.var.**State**(*name: str | None = None, tex_name: str | None = None, info: str | None = None, unit: str | None = None, v_str: str | float | None = None, v_iter: str | None = None, e_str: str | None = None, discrete: Discrete | None = None, t_const: BaseParam | DummyValue | BaseService | None = None, check_init: bool | None = True, v_setter: bool | None = False, e_setter: bool | None = False, addressable: bool | None = True, export: bool | None = True, diag_eps: float | None = 0.0, deps: List | None = None*)

Differential variable class, an alias of the *BaseVar*.

Parameters

t_const
[BaseParam, DummyValue] Left-hand time constant for the differential equation. They will be collected to array `dae.Tf`. Time constants will not be used when evaluating the right-hand side specified in `e_str` but will be applied to the left-hand side.

check_init
[bool] True to check if the equation right-hand-side is zero initially. Disabling the checking can be used for integrators when the initial input may not be zero.

Attributes

e_code
[str] Equation code string, equals string literal `f`

v_code
[str] Variable code string, equals string literal `x`

Examples

To implement the swing equation

$$M\dot{\omega} = \tau_m - \tau_e - D(\omega - 1)$$

Do the following in the `__init__()` of a model class:

```
self.omega = State(e_str = 'tm - te - D * (omega - 1)',
                   t_const = self.M,
                   ...
                   )
```

Note that `self.M`, the inertia parameter is given through `t_const` and is not part of `e_str`.

__init__(*name: str | None = None, tex_name: str | None = None, info: str | None = None, unit: str | None = None, v_str: str | float | None = None, v_iter: str | None = None, e_str: str | None = None, discrete: Discrete | None = None, t_const: BaseParam | DummyValue | BaseService | None = None, check_init: bool | None = True, v_setter: bool | None = False, e_setter: bool | None = False, addressable: bool | None = True, export: bool | None = True, diag_eps: float | None = 0.0, deps: List | None = None*)

Methods

`get_names()`	
`reset()`	Reset the internal numpy arrays and flags.
`set_address`(addr[, contiguous])	Set the address of internal variables.
`set_arrays`(dae[, inplace, alloc])	Set the equation and values arrays.

Attributes

`class_name`
`e_code`
`v_code`

andes.core.var.Algeb

class andes.core.var.**Algeb**(*name: str | None = None, tex_name: str | None = None, info: str | None = None, unit: str | None = None, v_str: str | float | None = None, v_iter: str | None = None, e_str: str | None = None, discrete: Discrete | None = None, v_setter: bool | None = False, e_setter: bool | None = False, v_str_add: bool | None = False, addressable: bool | None = True, export: bool | None = True, diag_eps: float | None = 0.0, deps: List | None = None, is_output: bool | None = False*)

Algebraic variable class, an alias of `andes.core.var.BaseVar`.

Note that residual equations corresponding to algebraic variables are given in an implicit form.

Attributes

e_code
[str] Equation code string, equals string literal `g`

v_code
[str] Variable code string, equals string literal `y`

Examples

When an algebraic variable `y` and the equation `y = x + z` shall be defined, use

```
e_str = 'x + z - y'
```

because it expresses the equation `x + z - y = 0`. It is a common mistake to use `e_str = 'x + z'`, which will result in a singular Jacobian matrix because `d(x + z) / d(y)` is zero.

__init__(*name: str | None = None, tex_name: str | None = None, info: str | None = None, unit: str | None = None, v_str: str | float | None = None, v_iter: str | None = None, e_str: str | None = None, discrete: Discrete | None = None, v_setter: bool | None = False, e_setter: bool | None = False, v_str_add: bool | None = False, addressable: bool | None = True, export: bool | None = True, diag_eps: float | None = 0.0, deps: List | None = None, is_output: bool | None = False*)

Methods

`get_names()`	
`reset()`	Reset the internal numpy arrays and flags.
`set_address(addr[, contiguous])`	Set the address of internal variables.
`set_arrays(dae[, inplace, alloc])`	Set the equation and values arrays.

Attributes

`class_name`
`e_code`
`v_code`

andes.core.var.ExtVar

class andes.core.var.**ExtVar**(*model: str, src: str, indexer: List | ndarray | BaseParam | BaseService | None = None, allow_none: bool | None = False, name: str | None = None, tex_name: str | None = None, ename: str | None = None, tex_ename: str | None = None, info: str | None = None, unit: str | None = None, v_str: str | float | None = None, v_iter: str | None = None, e_str: str | None = None, v_setter: bool | None = False, e_setter: bool | None = False, addressable: bool | None = True, export: bool | None = True, diag_eps: float | None = 0.0, is_input: bool | None = False*)

Algebraic variable that links to an external model.

This class is used to retrieve the addresses of a variable defined in an external model. An equation can be defined for the `ExtVar`. The evaluated value for the equation will be stored in the `ExtVar.e` attribute and added to the equations corresponding to the external variables.

Parameters

model
[str] Name of the source model

src
[str] Source variable name

indexer
[BaseParam, BaseService] A parameter of the hosting model, used as indices into the source model and variable. If is None, the source variable address will be fully copied.

allow_none
[bool, optional, default=False] True to allow None in indexer

e_str
[string, optional, default=None] Equation string, the evaluated value of which will be added to the source residual equation

Attributes

parent_model
[Model] The parent model providing the original parameter.

uid
[array-like] An array containing the absolute indices into the parent_instance values.

e_code
[str] Equation code string; copied from the parent instance.

v_code
[str] Variable code string; copied from the parent instance.

`__init__`(*model: str*, *src: str*, *indexer: List | ndarray | BaseParam | BaseService | None = None*, *allow_none: bool | None = False*, *name: str | None = None*, *tex_name: str | None = None*, *ename: str | None = None*, *tex_ename: str | None = None*, *info: str | None = None*, *unit: str | None = None*, *v_str: str | float | None = None*, *v_iter: str | None = None*, *e_str: str | None = None*, *v_setter: bool | None = False*, *e_setter: bool | None = False*, *addressable: bool | None = True*, *export: bool | None = True*, *diag_eps: float | None = 0.0*, *is_input: bool | None = False*)

Methods

`get_names()`	
`link_external`(ext_model)	Update variable addresses provided by external models
`reset()`	Reset the internal numpy arrays and flags.
`set_address`(addr[, contiguous])	Assigns address for equation RHS.
`set_arrays`(dae[, inplace, alloc])	Access `dae.h` or `dae.i` for the RHS of external variables when `e_str` exists..

Attributes

`class_name`

andes.core.var.ExtState

class andes.core.var.**ExtState**(*model: str, src: str, indexer: List | ndarray | BaseParam | BaseService | None = None, allow_none: bool | None = False, name: str | None = None, tex_name: str | None = None, ename: str | None = None, tex_ename: str | None = None, info: str | None = None, unit: str | None = None, v_str: str | float | None = None, v_iter: str | None = None, e_str: str | None = None, v_setter: bool | None = False, e_setter: bool | None = False, addressable: bool | None = True, export: bool | None = True, diag_eps: float | None = 0.0, is_input: bool | None = False*)

External state variable type.

> **Warning:** `ExtState` is not allowed to set `t_const`, as it may conflict with the source `State` variable.
>
> Only in rare cases should one set `e_str` for `ExtState`. The `t_const` of the source State variable is used.

__init__(*model: str, src: str, indexer: List | ndarray | BaseParam | BaseService | None = None, allow_none: bool | None = False, name: str | None = None, tex_name: str | None = None, ename: str | None = None, tex_ename: str | None = None, info: str | None = None, unit: str | None = None, v_str: str | float | None = None, v_iter: str | None = None, e_str: str | None = None, v_setter: bool | None = False, e_setter: bool | None = False, addressable: bool | None = True, export: bool | None = True, diag_eps: float | None = 0.0, is_input: bool | None = False*)

Methods

`get_names()`	
`link_external`(ext_model)	Update variable addresses provided by external models
`reset()`	Reset the internal numpy arrays and flags.
`set_address`(addr[, contiguous])	Assigns address for equation RHS.
`set_arrays`(dae[, inplace, alloc])	Access `dae.h` or `dae.i` for the RHS of external variables when `e_str` exists..

Attributes

`class_name`
`e_code`
`r_code`
`t_const`
`v_code`

andes.core.var.ExtAlgeb

class andes.core.var.**ExtAlgeb**(*model: str, src: str, indexer: List | ndarray | BaseParam | BaseService | None = None, allow_none: bool | None = False, name: str | None = None, tex_name: str | None = None, ename: str | None = None, tex_ename: str | None = None, info: str | None = None, unit: str | None = None, v_str: str | float | None = None, v_iter: str | None = None, e_str: str | None = None, v_setter: bool | None = False, e_setter: bool | None = False, addressable: bool | None = True, export: bool | None = True, diag_eps: float | None = 0.0, is_input: bool | None = False*)

External algebraic variable type.

__init__(*model: str, src: str, indexer: List | ndarray | BaseParam | BaseService | None = None, allow_none: bool | None = False, name: str | None = None, tex_name: str | None = None, ename: str | None = None, tex_ename: str | None = None, info: str | None = None, unit: str | None = None, v_str: str | float | None = None, v_iter: str | None = None, e_str: str | None = None, v_setter: bool | None = False, e_setter: bool | None = False, addressable: bool | None = True, export: bool | None = True, diag_eps: float | None = 0.0, is_input: bool | None = False*)

Methods

`get_names()`	
`link_external`(ext_model)	Update variable addresses provided by external models
`reset()`	Reset the internal numpy arrays and flags.
`set_address`(addr[, contiguous])	Assigns address for equation RHS.
`set_arrays`(dae[, inplace, alloc])	Access `dae.h` or `dae.i` for the RHS of external variables when `e_str` exists..

Attributes

`class_name`
`e_code`
`r_code`
`v_code`

andes.core.var.AliasState

class `andes.core.var.`**`AliasState`**(*var*, ***kwargs*)

Alias state variable.

Refer to the docs of `AliasAlgeb`.

`__init__`(*var*, ***kwargs*)

Methods

`get_names()`	
`link_external`(ext_model)	Update variable addresses provided by external models
`reset()`	Reset the internal numpy arrays and flags.
`set_address`(addr[, contiguous])	Assigns address for equation RHS.
`set_arrays`(dae[, inplace, alloc])	Access `dae.h` or `dae.i` for the RHS of external variables when `e_str` exists..

Attributes

`class_name`
`e_code`
`r_code`
`t_const`
`v_code`

andes.core.var.AliasAlgeb

class `andes.core.var.`**`AliasAlgeb`**(*var*, ***kwargs*)

Alias algebraic variable. Essentially `ExtAlgeb` that links to a a model's own variable.

`AliasAlgeb` is useful when the final output of a model is from a block, but the model must provide the final output in a pre-defined name. Using `AliasAlgeb`, A model can avoid adding an additional variable with a dummy equations.

Like `ExtVar`, labels of `AliasAlgeb` will not be saved in the final output. When plotting from file, one need to look up the original variable name.

`__init__`(*var*, ***kwargs*)

Methods

`get_names()`	
`link_external`(ext_model)	Update variable addresses provided by external models
`reset()`	Reset the internal numpy arrays and flags.
`set_address`(addr[, contiguous])	Assigns address for equation RHS.
`set_arrays`(dae[, inplace, alloc])	Access `dae.h` or `dae.i` for the RHS of external variables when `e_str` exists..

Attributes

`class_name`
`e_code`
`r_code`
`v_code`

`Observable`([e_str, name, tex_name, info, ...])	A variable computed by explicit assignment rather than solved in the DAE system.

andes.core.observable.Observable

class andes.core.observable.**Observable**(*e_str: str | None = None, name: str | None = None, tex_name: str | None = None, info: str | None = None, unit: str | None = None, discrete=None*)

A variable computed by explicit assignment rather than solved in the DAE system.

At code generation time, the Observable's expression is substituted into all referencing equations so that Jacobians are correctly computed via direct differentiation. Post-solve, values are evaluated and stored in `dae.b` for recording.

Parameters

e_str
[str, optional] Assignment expression string. Unlike `Algeb` where `e_str` means residual (`0 = e_str`), for Observable it means direct assignment (`b = e_str`).

Attributes

e_code
[None] Observable has no equation residual.

v_code
[str] Variable code string, equals string literal `b`.

Examples

To define an observable `vd` computed from bus voltage and angle:

```
self.vd = Observable(e_str='v * cos(delta - a)',
                     info='d-axis voltage',
                     tex_name='V_d')
```

The symbol `vd` can then be used in other equations. At code generation time, `vd` will be replaced by `v * cos(delta - a)` in those equations.

__init__(*e_str: str | None = None, name: str | None = None, tex_name: str | None = None, info: str | None = None, unit: str | None = None, discrete=None*)

Methods

`get_names()`	
`reset()`	Reset the internal numpy arrays and flags.
`set_address`(addr[, contiguous])	Set the address of this Observable in `dae.b`.
`set_arrays`(dae[, inplace, alloc])	Set the value array for this Observable.

Attributes

`class_name`
`e_code`
`v_code`

State

State variables are described by differential equations and can only change continuously. Use `State` for dynamics that evolve over time.

```
from andes.core.var import State

class Generator(Model):
    def __init__(self):
        # Rotor angle
        self.delta = State(
            v_str='delta0',            # Initial value
            e_str='omega - 1',         # d(delta)/dt = omega - 1
            info='Rotor angle',
            tex_name=r'\delta'
        )

        # Rotor speed
        self.omega = State(
            v_str='1.0',
            e_str='(Tm - Te - D*(omega-1)) / M',
            info='Rotor speed',
            tex_name=r'\omega'
        )
```

class andes.core.var.**State**(*name: str | None = None, tex_name: str | None = None, info: str | None = None, unit: str | None = None, v_str: str | float | None = None, v_iter: str | None = None, e_str: str | None = None, discrete: Discrete | None = None, t_const: BaseParam | DummyValue | BaseService | None = None, check_init: bool | None = True, v_setter: bool | None = False, e_setter: bool | None = False, addressable: bool | None = True, export: bool | None = True, diag_eps: float | None = 0.0, deps: List | None = None*)

Differential variable class, an alias of the *BaseVar*.

Parameters

t_const
[BaseParam, DummyValue] Left-hand time constant for the differential equation. They will be collected to array `dae.Tf`. Time constants will not be used when evaluating the right-hand side specified in `e_str` but will be applied to the left-hand side.

check_init
[bool] True to check if the equation right-hand-side is zero initially. Disabling the checking can be used for integrators when the initial input may not be zero.

Attributes

e_code
[str] Equation code string, equals string literal `f`

v_code
[str] Variable code string, equals string literal `x`

Examples

To implement the swing equation

$$M\dot{\omega} = \tau_m - \tau_e - D(\omega - 1)$$

Do the following in the `__init__()` of a model class:

```
self.omega = State(e_str = 'tm - te - D * (omega - 1)',
                   t_const = self.M,
                   ...
                   )
```

Note that `self.M`, the inertia parameter is given through `t_const` and is not part of `e_str`.

Algeb

Algebraic variables satisfy instantaneous constraints and can be discontinuous. Use `Algeb` for power balance equations, output calculations, and constraints.

```
from andes.core.var import Algeb

class Generator(Model):
    def __init__(self):
        # Electrical power output
        self.Pe = Algeb(
            v_str='p0',
            e_str='vd*Id + vq*Iq - Pe',
            info='Electrical power',
            tex_name='P_e'
        )
```

class andes.core.var.**Algeb**(*name: str | None = None, tex_name: str | None = None, info: str | None = None, unit: str | None = None, v_str: str | float | None = None, v_iter: str | None = None, e_str: str | None = None, discrete: Discrete | None = None, v_setter: bool | None = False, e_setter: bool | None = False, v_str_add: bool | None = False, addressable: bool | None = True, export: bool | None = True, diag_eps: float | None = 0.0, deps: List | None = None, is_output: bool | None = False*)

Algebraic variable class, an alias of `andes.core.var.BaseVar`.

Note that residual equations corresponding to algebraic variables are given in an implicit form.

Attributes

e_code
[str] Equation code string, equals string literal `g`

v_code
[str] Variable code string, equals string literal `y`

Examples

When an algebraic variable `y` and the equation `y = x + z` shall be defined, use

```
e_str = 'x + z - y'
```

because it expresses the equation `x + z - y = 0`. It is a common mistake to use `e_str = 'x + z'`, which will result in a singular Jacobian matrix because `d(x + z) / d(y)` is zero.

External Variables

Some models have "public" variables that can be accessed by other models. For example, a `Bus` defines `v` for voltage magnitude. Each device attached to a particular bus needs to access the value and impose reactive power injection.

External variables link with existing variables from another model or group using `ExtAlgeb` or `ExtState`.

ExtAlgeb

```
from andes.core.var import ExtAlgeb

class PQ(Model):
    def __init__(self):
        # Access bus voltage
        self.v = ExtAlgeb(
            src='v',                # Source variable name
            model='Bus',            # Source model
            indexer=self.bus,       # IdxParam for lookup
            e_str='-q',             # Inject reactive power
        )
```

class andes.core.var.**ExtAlgeb**(*model: str, src: str, indexer: List | ndarray | BaseParam | BaseService | None = None, allow_none: bool | None = False, name: str | None = None, tex_name: str | None = None, ename: str | None = None, tex_ename: str | None = None, info: str | None = None, unit: str | None = None, v_str: str | float | None = None, v_iter: str | None = None, e_str: str | None = None, v_setter: bool | None = False, e_setter: bool | None = False, addressable: bool | None = True, export: bool | None = True, diag_eps: float | None = 0.0, is_input: bool | None = False*)

External algebraic variable type.

ExtState

```
from andes.core.var import ExtState

class Exciter(Model):
    def __init__(self):
        # Access generator field voltage
        self.vf = ExtState(
            src='vf',
            model='GENROU',
            indexer=self.syn,
            e_str='Efd - vf0',
        )
```

class andes.core.var.**ExtState**(*model: str, src: str, indexer: List | ndarray | BaseParam | BaseService | None = None, allow_none: bool | None = False, name: str | None = None, tex_name: str | None = None, ename: str | None = None, tex_ename: str | None = None, info: str | None = None, unit: str | None = None, v_str: str | float | None = None, v_iter: str | None = None, e_str: str | None = None, v_setter: bool | None = False, e_setter: bool | None = False, addressable: bool | None = True, export: bool | None = True, diag_eps: float | None = 0.0, is_input: bool | None = False*)

External state variable type.

> **Warning:** `ExtState` is not allowed to set `t_const`, as it may conflict with the source `State` variable.
>
> Only in rare cases should one set `e_str` for `ExtState`. The `t_const` of the source State variable is used.

Value and Equation Strings

The most important feature of the symbolic framework is allowing equations to be defined using strings. There are three types of strings for a variable:

v_str: Explicit Initialization

Equation string for **explicit** initialization in the form of `v = v_str(x, y)`. The expression evaluates directly into the initial value.

```
self.omega = State(v_str='1.0')  # Start at 1.0 pu
self.Pe = Algeb(v_str='p0')      # Start at initial power
```

v_iter: Implicit Initialization

Equation string for **implicit** initialization in the form of `v_iter(x, y) = 0`. All `v_iter` equations are solved numerically using the Newton-Krylov iterative method.

```
self.Efd = Algeb(v_iter='Vf - Efd')  # Solve: Vf - Efd = 0
```

e_str: Equation Definition

Equation string for the differential or algebraic equation:

- For `State`: right-hand side of $\dot{x} = f(x, y)$
- For `Algeb`: residual that should equal zero, $g(x, y) = 0$

```
# Differential: d(omega)/dt = (Tm - Te) / M
self.omega = State(e_str='(Tm - Te) / M')

# Algebraic: 0 = P_gen - P_load
self.P = Algeb(e_str='Pgen - Pload')
```

Device Status: `u` vs `ue`

Every model has a `u` parameter (own online status, from input data) and a `ue` service (effective status, accounting for parent devices). Use **`ue`** in `e_str` to ensure equations are zeroed out when the device or any of its ancestors (bus, generator, exciter, etc.) is offline:

```
# Correct: equation respects parent status
self.omega = State(e_str='ue * (Tm - Te - D*(omega-1)) / M')

# Wrong: ignores bus/parent offline status
self.omega = State(e_str='u * (Tm - Te - D*(omega-1)) / M')
```

`ue` is computed as the product of the device's own `u` and the effective status of all parents declared with `status_parent=True` (see *Parameters*). For example, a governor's `ue` is `u * generator.ue * bus.ue`.

Use `u` in `v_str` for initialization, since `v_str` is evaluated before status propagation runs.

Flags for Value Overwriting

Variables have special flags for handling value initialization and equation values. This is only relevant for public or external variables.

v_setter Flag

The `v_setter` flag indicates whether a particular variable instance sets the initial value. If `v_setter=False` (default), variable values of the same address are added. If `v_setter=True`, the variable will set the values in the DAE array to its value, overwriting any previous values.

Only one variable at the same address is allowed to have `v_setter=True`.

e_setter Flag

The `e_setter` flag indicates whether the equation associated with a variable sets the equation value rather than adding to it.

A v_setter Example

A Bus is allowed to default the initial voltage magnitude to 1 and the voltage phase angle to 0. If a PV device is connected to a Bus device, the PV should override the voltage initial value with its voltage set point.

In `Bus.__init__()`:

```
self.v = Algeb(v_str='1')
```

In `PV.__init__()`:

```
self.v0 = Param()
self.bus = IdxParam(model='Bus')

self.v = ExtAlgeb(src='v',
                  model='Bus',
                  indexer=self.bus,
                  v_str='v0',
                  v_setter=True)
```

An `ExtAlgeb` is defined to access `Bus.v` using indexer `self.bus`. The `v_str` sets the initial value to `v0`. During the variable initialization phase for `PV`, `PV.v.v` is set to `v0`.

During value collection into `DAE.y` by the `System` class, `PV.v`, as a final `v_setter`, will overwrite the voltage magnitude for Bus devices with the indices provided in `PV.bus`.

Observable

Observable variables are computed by explicit assignment rather than solved in the DAE system. At code generation time, the expression is substituted into all referencing equations so that Jacobians are computed correctly. Post-solve, values are evaluated and stored in `dae.b` for recording.

Use `Observable` instead of `Algeb` when a variable is a pure explicit function of other variables and does not need to participate in the Newton-Raphson solve. This reduces the DAE system size without sacrificing Jacobian accuracy.

```
from andes.core.observable import Observable

class MyModel(Model):
    def __init__(self):
        self.vd = Observable(
            e_str='v * cos(delta - a)',
            info='d-axis voltage',
            tex_name='V_d',
        )
```

Note: Observable variables cannot be referenced via `ExtAlgeb`. If another model needs to link to the variable externally, keep it as `Algeb`.

class andes.core.observable.**Observable**(*e_str: str | None = None, name: str | None = None, tex_name: str | None = None, info: str | None = None, unit: str | None = None, discrete=None*)

A variable computed by explicit assignment rather than solved in the DAE system.

At code generation time, the Observable's expression is substituted into all referencing equations so that Jacobians are correctly computed via direct differentiation. Post-solve, values are evaluated and stored in `dae.b` for recording.

Parameters

e_str
[str, optional] Assignment expression string. Unlike `Algeb` where `e_str` means residual (`0 = e_str`), for Observable it means direct assignment (`b = e_str`).

Attributes

e_code
[None] Observable has no equation residual.

v_code
[str] Variable code string, equals string literal `b`.

Examples

To define an observable `vd` computed from bus voltage and angle:

```
self.vd = Observable(e_str='v * cos(delta - a)',
                     info='d-axis voltage',
                     tex_name='V_d')
```

The symbol `vd` can then be used in other equations. At code generation time, `vd` will be replaced by `v * cos(delta - a)` in those equations.

reset()

Reset the internal numpy arrays and flags.

set_address(*addr: ndarray*, *contiguous=False*)

Set the address of this Observable in `dae.b`.

Overrides BaseVar to ensure `e_inplace` is always False (no equation array for Observable).

set_arrays(*dae*, *inplace=True*, *alloc=True*)

Set the value array for this Observable.

Only sets `self.v` (into `dae.b`). No equation array is allocated since Observable has no residual.

Alias Variables

Create references to existing variables for convenience:

```
from andes.core.var import AliasState, AliasAlgeb

# Reference generator speed
self.omega = AliasState(self.GENROU.omega)
```

class andes.core.var.**AliasState**(*var*, ***kwargs*)

Alias state variable.

Refer to the docs of `AliasAlgeb`.

class andes.core.var.**AliasAlgeb**(*var*, ***kwargs*)

Alias algebraic variable. Essentially `ExtAlgeb` that links to a a model's own variable.

`AliasAlgeb` is useful when the final output of a model is from a block, but the model must provide the final output in a pre-defined name. Using `AliasAlgeb`, A model can avoid adding an additional variable with a dummy equations.

Like `ExtVar`, labels of `AliasAlgeb` will not be saved in the final output. When plotting from file, one need to look up the original variable name.

BaseVar

The abstract base class defining the variable interface.

class andes.core.var.**BaseVar**(*name: str | None = None, tex_name: str | None = None, info: str | None = None, unit: str | None = None, v_str: str | float | None = None, v_iter: str | None = None, e_str: str | None = None, discrete: Discrete | None = None, v_setter: bool | None = False, e_setter: bool | None = False, v_str_add: bool | None = False, addressable: bool | None = True, export: bool | None = True, diag_eps: float | None = 0.0, deps: List | None = None, is_output: bool | None = False*)

Base variable class.

Derived classes *State* and *Algeb* should be used to build model variables.

Parameters

info
[str, optional] Descriptive information

unit
[str, optional] Unit

tex_name
[str] LaTeX-formatted variable symbol. If is None, the value of *name* will be used.

discrete
[Discrete] Discrete component on which this variable depends. ANDES will call *check_var()* of the discrete component before initializing this variable.

name
[str, optional] Variable name. One should typically assigning the name directly because it will be automatically assigned by the model. The value of `name` will be the symbol name to be used in expressions.

Attributes

a
[array-like] variable address

v
[array-like] local-storage of the variable value

e
[array-like] local-storage of the corresponding equation value

e_str
[str] the string/symbolic representation of the equation

v_str
[str] explicit initialization equation

v_str_add
[bool] True if the value of *v_str* will be added to the variable. Useful when other models access this variable and set part of the initial value

v_iter
[str] implicit iterative equation in the form of 0 = v_iter

reset()

Reset the internal numpy arrays and flags.

set_address(*addr: ndarray*, *contiguous=False*)

Set the address of internal variables.

Parameters

addr
[np.ndarray] The assigned address for this variable

contiguous
[bool, optional] If the addresses are contiguous

set_arrays(*dae*, *inplace=True*, *alloc=True*)

Set the equation and values arrays.

Parameters

dae
[DAE] Reference to System.dae

Common Patterns

Power Balance

```
# Algebraic equation: 0 = P_gen - P_load - P_line
self.P = Algeb(e_str='Pgen - Pload - Pline')
```

Swing Equation

```
self.delta = State(e_str='omega - 1')
self.omega = State(e_str='(Tm - Te - D*(omega-1)) / M')
```

External Power Injection

```
# Inject power to bus (negative = generation/injection)
self.a = ExtAlgeb(src='a', model='Bus', indexer=self.bus,
                  e_str='-p')
self.v = ExtAlgeb(src='v', model='Bus', indexer=self.bus,
                  e_str='-q')
```

See Also

- *Atomic Types* - v-provider and e-provider concepts
- *DAE Formulation* - DAE mathematical background
- *Parameters* - Parameter types
- *Services* - Intermediate calculations

4.3.3 Services

Services are helper variables outside the DAE variable list. They are most often used for storing intermediate constants but can be used for special operations to work around restrictions in the symbolic framework.

Background

Services are **v-providers**, meaning each service has an attribute `v` for storing service values. Unlike variables, services do not participate in the DAE system directly but provide computed values that variables and equations can use.

The base class of services is `andes.core.service.BaseService`.

Service Types

`BaseService`([name, tex_name, unit, info, vtype])	Base class for Service.
`OperationService`([name, tex_name, info])	Base class for a type of Service which performs specific operations.
`ConstService`([v_str, v_numeric, vtype, ...])	A type of Service that stays constant once initialized.
`VarService`([v_str, v_numeric, vtype, name, ...])	Variable service that gets updated in each step/iteration before computing the residual equations.
`ExtService`(model, src, indexer[, attr, ...])	Service constants whose value is from an external model or group.
`PostInitService`([v_str, v_numeric, vtype, ...])	Constant service that gets stored once after init.
`NumReduce`(u, ref, fun[, name, tex_name, ...])	A helper Service type which reduces a linearly stored 2-D ExtParam into 1-D Service.
`NumRepeat`(u, ref, **kwargs)	A helper Service type which repeats a v-provider's value based on the shape from a BackRef
`IdxRepeat`(u, ref, **kwargs)	Helper class to repeat IdxParam.
`BackRef`(**kwargs)	A special type of reference collector.
`RefFlatten`(ref, **kwargs)	A service type for flattening `andes.core.service.BackRef` into a 1-D list.
`EventFlag`(u[, vtype, name, tex_name, info])	Service to flag events when the input value changes.
`VarHold`(u, hold[, vtype, name, tex_name, info])	Service for holding the input when the hold signal is on.
`ExtendedEvent`(u[, t_ext, trig, enable, ...])	Service for indicating an event for an extended, predefined period of time following the event disappearance.
`DataSelect`(optional, fallback[, name, ...])	Class for selecting values for optional DataParam or NumParam.
`NumSelect`(optional, fallback, name, ...)	Class for selecting values for optional NumParam.
`DeviceFinder`(u, link, idx_name, default_model)	Service for finding `idx` of devices which are linked to the given devices.
`FlagCondition`(u, func[, flag, name, ...])	Class for flagging values based on a condition function.
`FlagValue`(u, value[, flag, name, tex_name, ...])	Class for flagging values that equal to the given value.
`FlagGreaterThan`(u[, value, flag, equal, ...])	Service for flagging parameters > or >= the given value element-wise.
`FlagLessThan`(u[, value, flag, equal, name, ...])	Service for flagging parameters < or <= the given value element-wise.
`InitChecker`(u[, lower, upper, equal, ...])	Class for checking init values against known typical values.
`Replace`(old_val, flt, new_val[, name, ...])	Replace parameters with new values if the function returns True
`ApplyFunc`(u, func[, name, tex_name, info, cache])	Class for applying a numerical function on a parameter..

andes.core.service.BaseService

class andes.core.service.**BaseService**(*name: str = None, tex_name: str = None, unit: str = None, info: str = None, vtype: Type = None*)

Base class for Service.

Service is a v-provider type for holding internal and temporary values. Subclasses need to implement `v` as a member attribute or using a property decorator.

Parameters

name
[str] Instance name

Attributes

owner
[Model] The hosting/owner model instance

__init__(*name: str = None, tex_name: str = None, unit: str = None, info: str = None, vtype: Type = None*)

Methods

`assign_memory`(n)	Assign memory for `self.v` and set the array to zero.
`get_names`()	Return *name* in a list

Attributes

`class_name`	Return the class name
`n`	Return the count of values in `self.v`.

andes.core.service.OperationService

class andes.core.service.**OperationService**(*name=None, tex_name=None, info=None*)

Base class for a type of Service which performs specific operations. OperationService may not use the *assign_memory* from *BaseService*, because it can have a different size.

This class cannot be used by itself.

See also:

NumReduce
Service for Reducing linearly stored 2-D services into 1-D

NumRepeat
Service for repeating 1-D NumParam/ v-array following a

sub-pattern
IdxRepeat
Service for repeating 1-D IdxParam/ v-list following a

sub-pattern

__init__(*name=None*, *tex_name=None*, *info=None*)

Methods

`assign_memory`(n)	Assign memory for `self.v` and set the array to zero.
`get_names`()	Return *name* in a list

Attributes

`class_name`	Return the class name
`n`	Return the count of values in `self.v`.
`v`	Return values stored in *self._v*.

andes.core.service.ConstService

class andes.core.service.**ConstService**(*v_str: str | None = None*, *v_numeric: Callable | None = None*, *vtype: type | None = None*, *name: str | None = None*, *tex_name: str | None = None*, *info: str | None = None*, *unit: str | None = None*)

A type of Service that stays constant once initialized.

ConstService are usually constants calculated from parameters. They are only evaluated once in the initialization phase before variables are initialized. Therefore, uninitialized variables must not be used in *v_str*`.

ConstService are evaluated *in sequence* after getting external variables and parameters and before initializing internal variables.

Parameters

name
[str] Name of the ConstService

v_str
[str] An equation string to calculate the variable value.

v_numeric
[Callable, optional] A callable which returns the value of the ConstService

v_type: type, optional, default to float
Type of element in the value array in float or complex

Attributes

v
[array-like or a scalar] ConstService value

_v_t0
[np.ndarray or None] `v` at t=0, saved by `snapshot_init()`. Used by `restore_init()` to reset `v` to the post-init baseline.

`__init__`(*v_str: str | None = None, v_numeric: Callable | None = None, vtype: type | None = None, name: str | None = None, tex_name: str | None = None, info: str | None = None, unit: str | None = None*)

Methods

`assign_memory`(n)	Assign memory for `self.v` and set the array to zero.
`get_names`()	Return *name* in a list
`restore_init`()	Restore `v` from `_v_t0` saved by `snapshot_init()`.
`snapshot_init`()	Save the current `v` into `_v_t0` for `restore_init()`.

Attributes

`class_name`	Return the class name
`n`	Return the count of values in `self.v`.

andes.core.service.VarService

class andes.core.service.**VarService**(*v_str: str | None = None, v_numeric: Callable | None = None, vtype: type | None = None, name: str | None = None, tex_name: str | None = None, info: str | None = None, unit: str | None = None, sequential: bool | None = True*)

Variable service that gets updated in each step/iteration before computing the residual equations. As a results, variable values from the k-th step are used to compute a `VarService` that will be used to compute the residual for the (k+1)-th step.

This class is useful when one has non-differentiable algebraic equations, which make use of *abs()*, *re* and *im*. Instead of creating *Algeb*, one can put the equation in *VarService*, which will be updated before solving algebraic equations.

Parameters

sequential
[bool, optional, default to True] True if this VarService depends on previously defined VarService and should be evaluated in sequence. False if this VarService only uses known variables.

Warning: `VarService` is intended for non-differentiable expressions (e.g., `Abs()`, `re()`, `im()`) that cannot appear in algebraic equations. If the expression is differentiable, an `Algeb` variable should be used instead.

During model initialization, `VarService` is evaluated *before* any variable `v_str` assignment is computed. If the expression references algebraic or state variables, those variables will still hold their default (typically zero) values, resulting in an incorrect initial `VarService` value. Variables whose `v_str` depends on this `VarService` will consequently be initialized from an erroneous starting point. At runtime, this effect is self-correcting because `VarService` is re-evaluated every iteration with updated variable values. However, the incorrect initial value can cause convergence difficulties or incorrect limiter flag settings during the first few iterations.

`VarService` is not solved simultaneously with algebraic equations, meaning that a one-step delay exists between the algebraic variables and the `VarService` value.

Examples

In ESST3A model, the voltage and current sensors (vd + jvq), (Id + jIq) estimate the sensed VE using equation

$$VE = |K_{PC} * (v_d + 1jv_q) + 1j(K_I + K_{PC} * X_L) * (I_d + 1jI_q)|$$

One can use *VarService* to implement this equation

```
self.VE = VarService(
    tex_name='V_E', info='VE', v_str='Abs(KPC*(vd + 1j*vq) + 1j*(KI +
    KPC*XL)*(Id + 1j*Iq))', )
```

__init__(*v_str: str | None = None, v_numeric: Callable | None = None, vtype: type | None = None, name: str | None = None, tex_name: str | None = None, info: str | None = None, unit: str | None = None, sequential: bool | None = True*)

Methods

`assign_memory(n)`	Assign memory for `self.v` and set the array to zero.
`get_names()`	Return *name* in a list
`restore_init()`	No-op: VarService is recomputed each step.
`snapshot_init()`	No-op: VarService is recomputed each step.

Attributes

`class_name`	Return the class name
`n`	Return the count of values in `self.v`.

andes.core.service.ExtService

class andes.core.service.**ExtService**(*model: str, src: str, indexer:* BaseParam | BaseService, *attr: str = 'v', allow_none: bool = False, default=0, name: str = None, tex_name: str = None, vtype=None, info: str = None*)

Service constants whose value is from an external model or group.

Parameters

src
[str] Variable or parameter name in the source model or group

model
[str] A model name or a group name

indexer
[IdxParam or BaseParam] An "Indexer" instance whose `v` field contains the `idx` of devices in the model or group.

Examples

A synchronous generator needs to retrieve the `p` and `q` values from static generators for initialization. `ExtService` is used for this purpose.

In a synchronous generator, one can define the following to retrieve `StaticGen.p` as `p0`:

```
class GENCLSModel(Model):
    def __init__(...):
        ...
        self.p0 = ExtService(src='p',
```

(continues on next page)

(continued from previous page)

```
                                    model='StaticGen',
                                    indexer=self.gen,
                                    tex_name='P_0')
```

__init__(*model: str, src: str, indexer: BaseParam | BaseService, attr: str = 'v', allow_none: bool = False, default=0, name: str = None, tex_name: str = None, vtype=None, info: str = None*)

Methods

`assign_memory`(n)	Assign memory for `self.v` and set the array to zero.
`get_names`()	Return *name* in a list
`link_external`(ext_model)	Method to be called by `System` for getting values from the external model or group.

Attributes

`class_name`	Return the class name
`n`	Return the count of values in `self.v`.

andes.core.service.PostInitService

class andes.core.service.**PostInitService**(*v_str: str | None = None, v_numeric: Callable | None = None, vtype: type | None = None, name: str | None = None, tex_name: str | None = None, info: str | None = None, unit: str | None = None*)

Constant service that gets stored once after init.

This service is useful when one need to store initialization values stored in variables.

Examples

In ESST3A model, the *vf* variable is initialized followed by other variables. One can store the initial *vf* into *vf0* so that equation `vf - vf0 = 0` will hold.

```
self.vref0 = PostInitService(info='Initial reference voltage input',
                             tex_name='V_{ref0}', v_str='vref', )
```

Since all *ConstService* are evaluated before equation evaluation, without using PostInitService, one will need to create lots of *ConstService* to store values in the initialization path towards *vf0*, in order to correctly initialize *vf*.

__init__(*v_str: str | None = None, v_numeric: Callable | None = None, vtype: type | None = None, name: str | None = None, tex_name: str | None = None, info: str | None = None, unit: str | None = None*)

Methods

`assign_memory`(n)	Assign memory for `self.v` and set the array to zero.
`get_names`()	Return *name* in a list
`restore_init`()	Restore `v` from `_v_t0` saved by `snapshot_init()`.
`snapshot_init`()	Save the current `v` into `_v_t0` for `restore_init()`.

Attributes

`class_name`	Return the class name
`n`	Return the count of values in `self.v`.

andes.core.service.NumReduce

class andes.core.service.**NumReduce**(*u, ref:* BackRef, *fun: Callable, name=None, tex_name=None, info=None, cache=True*)

A helper Service type which reduces a linearly stored 2-D ExtParam into 1-D Service.

NumReduce works with ExtParam whose *v* field is a list of lists. A reduce function which takes an array-like and returns a scalar need to be supplied. NumReduce calls the reduce function on each of the lists and return all the scalars in an array.

Parameters

u
[ExtParam] Input ExtParam whose `v` contains linearly stored 2-dimensional values

ref
[BackRef] The BackRef whose 2-dimensional shapes are used for indexing

fun
[Callable] The callable for converting a 1-D array-like to a scalar

Examples

Suppose one wants to calculate the mean value of the `Vn` in one Area. In the `Area` class, one defines

```
class AreaModel(...):
    def __init__(...):
        ...
        # backward reference from `Bus`
        self.Bus = BackRef()

        # collect the Vn in an 1-D array
        self.Vn = ExtParam(model='Bus',
                           src='Vn',
                           indexer=self.Bus)

        self.Vn_mean = NumReduce(u=self.Vn,
                                 fun=np.mean,
                                 ref=self.Bus)
```

Suppose we define two areas, 1 and 2, the Bus data looks like

idx	area	Vn
1	1	110
2	2	220
3	1	345
4	1	500

Then, *self.Bus.v* is a list of two lists `[ [1, 3, 4], [2] ]`. *self.Vn.v* will be retrieved and linearly stored as `[110, 345, 500, 220]`. Based on the shape from *self.Bus*, `numpy.mean()` will be called on `[110, 345, 500]` and `[220]` respectively. Thus, *self.Vn_mean.v* will become `[318.33, 220]`.

__init__(*u*, *ref:* BackRef, *fun:* *Callable*, *name=None*, *tex_name=None*, *info=None*, *cache=True*)

Methods

`assign_memory`(n)	Assign memory for `self.v` and set the array to zero.
`get_names`()	Return *name* in a list

Attributes

class_name	Return the class name
n	Return the count of values in self.v.
v	Return the reduced values from the reduction function in an array

andes.core.service.NumRepeat

class andes.core.service.**NumRepeat**(*u*, *ref*, ***kwargs*)

A helper Service type which repeats a v-provider's value based on the shape from a BackRef

Examples

NumRepeat was originally designed for computing the inertia-weighted average rotor speed (center of inertia speed). COI speed is computed with

$$\omega_{COI} = \frac{\sum M_i * \omega_i}{\sum M_i}$$

The numerator can be calculated with a mix of BackRef, ExtParam and ExtState. The denominator needs to be calculated with NumReduce and Service Repeat. That is, use NumReduce to calculate the sum, and use NumRepeat to repeat the summed value for each device.

In the COI class, one would have

```
class COIModel(...):
    def __init__(...):
        ...
        self.SynGen = BackRef()
        self.SynGenIdx = RefFlatten(ref=self.SynGen)
        self.M = ExtParam(model='SynGen',
                          src='M',
                          indexer=self.SynGenIdx)

        self.wgen = ExtState(model='SynGen',
                             src='omega',
                             indexer=self.SynGenIdx)

        self.Mt = NumReduce(u=self.M,
                            fun=np.sum,
                            ref=self.SynGen)

        self.Mtr = NumRepeat(u=self.Mt,
                             ref=self.SynGen)

        self.pidx = IdxRepeat(u=self.idx,ref=self.SynGen)
```

Finally, one would define the center of inertia speed as

```
self.wcoi = Algeb(v_str='1', e_str='-wcoi')

self.wcoi_sub = ExtAlgeb(model='COI',
                         src='wcoi',
                         e_str='M * wgen / Mtr',
                         v_str='M / Mtr',
                         indexer=self.pidx,
                         )
```

It is very worth noting that the implementation uses a trick to separate the average weighted sum into n sub-equations, each calculating the $(M_i * \omega_i)/(\sum M_i)$. Since all the variables are preserved in the sub-equation, the derivatives can be calculated correctly.

__init__(*u*, *ref*, ***kwargs*)

Methods

`assign_memory`(n)	Assign memory for `self.v` and set the array to zero.
`get_names`()	Return *name* in a list

Attributes

`class_name`	Return the class name
`n`	Return the count of values in `self.v`.
`v`	Return the values of the repeated values in a sequential 1-D array

andes.core.service.IdxRepeat

class `andes.core.service.`**IdxRepeat**(*u*, *ref*, ***kwargs*)

Helper class to repeat IdxParam.

This class has the same functionality as *andes.core.service.NumRepeat* but only operates on IdxParam, DataParam or NumParam.

__init__(*u*, *ref*, ***kwargs*)

Methods

`assign_memory`(n)	Assign memory for `self.v` and set the array to zero.
`get_names`()	Return *name* in a list

Attributes

`class_name`	Return the class name
`n`	Return the count of values in `self.v`.
`v`	Return values stored in *self._v*.

andes.core.service.BackRef

class andes.core.service.**BackRef**(***kwargs*)

A special type of reference collector.

BackRef is used for collecting device indices of other models referencing the parent model of the *BackRef*. The *v``field will be a list of lists, each containing the `idx* of other models referencing each device of the parent model.

BackRef can be passed as indexer for params and vars, or shape for *NumReduce* and *NumRepeat*. See examples for illustration.

See also:

andes.core.service.NumReduce
A more complete example using BackRef to build the COI model

Examples

A Bus device has an *IdxParam* of *area*, storing the *idx* of area to which the bus device belongs. In `Bus.__init__()`, one has

```
self.area = IdxParam(model='Area')
```

Suppose *Bus* has the following data

idx	area	Vn
1	1	110
2	2	220
3	1	345
4	1	500

The Area model wants to collect the indices of Bus devices which points to the corresponding Area device. In `Area.__init__`, one defines

```
self.Bus = BackRef()
```

where the member attribute name *Bus* needs to match exactly model name that *Area* wants to collect *idx* for. Similarly, one can define `self.ACNode = BackRef()` to collect devices in the *ACNode* group that references Area.

The collection of *idx* happens in `andes.system.System._collect_ref_param()`. It has to be noted that the specific *Area* entry must exist to collect model idx-dx referencing it. For example, if *Area* has the following data

```
idx 1
```

Then, only Bus 1, 3, and 4 will be collected into *self.Bus.v*, namely, `self.Bus.v == [ [1, 3, 4] ]`.

If *Area* has data

```
idx 1 2
```

Then, *self.Bus.v* will end up with `[ [1, 3, 4], [2] ]`.

__init__(***kwargs*)

Methods

`assign_memory`(n)	Assign memory for `self.v` and set the array to zero.
`get_names`()	Return *name* in a list

Attributes

`class_name`	Return the class name
`n`	Return the count of values in `self.v`.

andes.core.service.RefFlatten

class andes.core.service.**RefFlatten**(*ref*, ***kwargs*)

A service type for flattening `andes.core.service.BackRef` into a 1-D list.

Examples

This class is used when one wants to pass *BackRef* values as indexer.

`andes.models.coi.COI` collects referencing `andes.models.group.SynGen` with

```
self.SynGen = BackRef(info='SynGen idx lists', export=False)
```

After collecting BackRefs, *self.SynGen.v* will become a two-level list of indices, where the first level correspond to each COI and the second level correspond to generators of the COI.

Convert *self.SynGen* into 1-d as *self.SynGenIdx*, which can be passed as indexer for retrieving other parameters and variables

```
self.SynGenIdx = RefFlatten(ref=self.SynGen)

self.M = ExtParam(model='SynGen', src='M',
                  indexer=self.SynGenIdx, export=False,
                  )
```

__init__(*ref*, ***kwargs*)

Methods

`assign_memory`(n)	Assign memory for `self.v` and set the array to zero.
`get_names`()	Return *name* in a list

Attributes

`class_name`	Return the class name
`n`	Return the count of values in `self.v`.
`v`	Return values stored in *self._v*.

andes.core.service.EventFlag

class andes.core.service.**EventFlag**(*u*, *vtype: type | None = None*, *name: str | None = None*, *tex_name=None*, *info=None*)

Service to flag events when the input value changes. The typical input is a *v-provider* with binary values.

Implemented by providing *self.check(**kwargs)* as *v_numeric*. *EventFlag.v* stores the values of the input variable in the most recent iteration/step.

After the evaluation of *self.check()*, *self.v* will be updated.

__init__(*u*, *vtype: type | None = None*, *name: str | None = None*, *tex_name=None*, *info=None*)

Methods

`assign_memory`(n)	Assign memory for `self.v` and set the array to zero.
`check`(**kwargs)	Check status and set event flags.
`get_names`()	Return *name* in a list
`restore_init`()	No-op: VarService is recomputed each step.
`snapshot_init`()	No-op: VarService is recomputed each step.

Attributes

`class_name`	Return the class name
`n`	Return the count of values in `self.v`.

andes.core.service.VarHold

class andes.core.service.**VarHold**(*u*, *hold*, *vtype=None*, *name=None*, *tex_name=None*, *info=None*)

Service for holding the input when the hold signal is on.

Parameters

hold

[v-provider, binary] Hold signal array with length equal to the input. For elements that are 1, the corresponding inputs are held until the hold signal returns to 0.

__init__(*u*, *hold*, *vtype=None*, *name=None*, *tex_name=None*, *info=None*)

Methods

`assign_memory`(n)	Assign memory for `self.v` and set the array to zero.
`check`(**kwargs)	Custom *v_numeric* function for checking the hold signal and calculating outputs.
`get_names`()	Return *name* in a list
`restore_init`()	No-op: VarService is recomputed each step.
`snapshot_init`()	No-op: VarService is recomputed each step.

Attributes

`class_name`	Return the class name
`n`	Return the count of values in `self.v`.

andes.core.service.ExtendedEvent

class andes.core.service.**ExtendedEvent**(*u*, *t_ext: int | float | BaseParam | BaseService = 0.0*, *trig: str = 'rise'*, *enable=True*, *v_disabled=0*, *extend_only=False*, *vtype: type | None = None*, *name: str | None = None*, *tex_name=None*, *info=None*)

Service for indicating an event for an extended, predefined period of time following the event disappearance.

The triggering of an event, whether the rise or fall edge, is specified through *trig*. For example, if *trig* = *rise*, the change of the input from 0 to 1 will be considered as an input, whereas the subsequent change back to 0 will be considered as the event end.

ExtendedEvent.v stores the flags whether the extended time has completed. Outputs will become 1 once the event starts and return to 0 when the extended time ends.

Parameters

u
[v-provider] Triggering signal where the values are 0 or 1.

trig
[str in ("rise", "fall")] Triggering edge for the beginning of an event. *rise* by default.

enable
[bool or v-provider] If disabled, the output will be *v_disabled*

extend_only
[bool] Only output during the extended period, not the event period.

> **Warning:** The performance of this class needs to be optimized.

__init__(*u*, *t_ext: int | float | BaseParam | BaseService = 0.0*, *trig: str = 'rise'*, *enable=True*, *v_disabled=0*, *extend_only=False*, *vtype: type | None = None*, *name: str | None = None*, *tex_name=None*, *info=None*)

Methods

`assign_memory`(n)	Assign memory for internal data.
`check`(**kwargs)	Check if an extended event is in place.
`get_names`()	Return *name* in a list
`restore_init`()	No-op: VarService is recomputed each step.
`snapshot_init`()	No-op: VarService is recomputed each step.

Attributes

`class_name`	Return the class name
`n`	Return the count of values in `self.v`.

andes.core.service.DataSelect

class andes.core.service.**DataSelect**(*optional*, *fallback*, *name: str | None = None*, *tex_name: str | None = None*, *info: str | None = None*)

Class for selecting values for optional DataParam or NumParam.

This service is a v-provider that uses optional DataParam when available. Otherwise, use the fallback value.

DataParam will be tested for *None*, and NumParam will be tested with `np.isnan()`.

Notes

An use case of DataSelect is remote bus. One can do

```
self.buss = DataSelect(option=self.busr, fallback=self.bus)
```

Then, pass `self.buss` instead of `self.bus` as indexer to retrieve voltages.

Another use case is to allow an optional turbine rating. One can do

```
self.Tn = NumParam(default=None)
self.Sg = ExtParam(...)
self.Sn = DataSelect(Tn, Sg)
```

__init__(*optional*, *fallback*, *name: str | None = None*, *tex_name: str | None = None*, *info: str | None = None*)

Methods

`assign_memory`(n)	Assign memory for `self.v` and set the array to zero.
`get_names`()	Return *name* in a list

Attributes

`class_name`	Return the class name
`n`	Return the count of values in `self.v`.
`v`	

andes.core.service.NumSelect

class andes.core.service.**NumSelect**(*optional*, *fallback*, *name: str | None = None*, *tex_name: str | None = None*, *info: str | None = None*, *ignore_cond: ~typing.Callable | None = functools.partial(<ufunc 'equal'>, 0)*)

Class for selecting values for optional NumParam.

NumSelect works with internal and external parameters.

Any values equal to `np.nan` will always be ignored. If one needs to ignore values based on additional conditions, pass it through `ignore_cond`. For example, to ignore zero values, use `ignore_cond = partial(np.equal, 0)`.

Examples

One use case is to allow an optional turbine rating. One can do

```
self.Tn = NumParam(default=None) self.Sg = ExtParam(...) self.Sn =
DataSelect(Tn, Sg)
```

__init__(*optional*, *fallback*, *name: str | None = None*, *tex_name: str | None = None*, *info: str | None = None*, *ignore_cond: ~typing.Callable | None = functools.partial(<ufunc 'equal'>, 0)*)

Methods

`assign_memory`(n)	Assign memory for `self.v` and set the array to zero.
`get_names`()	Return *name* in a list

Attributes

`class_name`	Return the class name
`n`	Return the count of values in `self.v`.
`v`	Return values stored in *self._v*.

andes.core.service.DeviceFinder

class andes.core.service.**DeviceFinder**(*u*, *link*, *idx_name: str*, *default_model: str*, *auto_find: bool | None = True*, *auto_add: bool | None = True*, *name: str | None = None*, *tex_name: str | None = None*, *info: str | None = None*)

Service for finding `idx` of devices which are linked to the given devices.

The `auto_find` parameter controls if the device idx should be automatically looked up. The `auto_add` parameter controls if the device will be automatically added. The two parameters are not exclusive. One can skip finding the device but automatically adding it.

If `auto_find` is `True` and the `idx` is None, DeviceFinder will look up for the device. If not found and `auto_add` is `True`, DevFinder will then automatically add the devices. The `idx` of the devices that are found or added will be stored to the DeviceFinder instance, so that *DeviceFinder* can be used like any *IdxParam*.

Adding new devices are called at the beginning of `andes.system.System.setup()`.

Examples

The IEEEST stabilizer takes an optional parameter `busf` of the type *IdxParam* for specifying the connected bus frequency measurement device, which is needed for mode 6. To avoid reimplementing *BusFreq* within IEEEST, one can do

```
self.busfreq = DeviceFinder(self.busf,
                            link=self.buss, idx_name='bus',
                            default_model='BusFreq')
```

where `self.busf` is for the optional parameter for the `idx` of bus frequency estimation devices (e.g., *BusFreq*), `self.buss` is for the `idx` of buses that `self.busf` devices should measure, and `idx_name` is the name of the BusFreq parameter through which the indices of measured buses are given.

For each `None` or invalid values in `self.busf`, a *BusFreq* device will be created with its `bus` set to the corresponding value in `self.buss`. That is, `BusFreq.[idx_name].v = [link]`.

At the end, the *DeviceFinder* instance will contain the list of `BusFreq` that are are connected to *self.buss*, respectively.

In the case of any valid value in `self.busf`, that is, the value is an existing `BusFreq` device, *DeviceFinder* will return it as is without checking if the `BusFreq` device actually measures the bus specified by `self.buss`. It allows to use the measurement at a different location, but the user have to perform the data consistency check.

`__init__`(*u*, *link*, *idx_name: str*, *default_model: str*, *auto_find: bool | None = True*, *auto_add: bool | None = True*, *name: str | None = None*, *tex_name: str | None = None*, *info: str | None = None*)

Methods

`assign_memory`(n)	Assign memory for `self.v` and set the array to zero.
`find_or_add`(system)	Find or add devices.
`get_names`()	Return *name* in a list

Attributes

`class_name`	Return the class name
`n`	Return the count of values in `self.v`.

andes.core.service.FlagCondition

class andes.core.service.**FlagCondition**(*u*, *func*, *flag=1*, *name=None*, *tex_name=None*, *info=None*, *cache=True*)

Class for flagging values based on a condition function.

By default, values whose condition function output equal that equal to True/1 will be flagged as *1*. *0* otherwise.

Parameters

u
Input parameter

func
A condition function that returns True or False.

flag
[1 by default, only 0 or 1 is accepted.] The flag for the inputs whose condition output is True.

> **Warning:** This class is not ready.
>
> *FlagCondition* can only be applied to *BaseParam* with *cache=True*. Applying to *Service* will fail unless *cache* is False (at a performance cost).

__init__(*u*, *func*, *flag=1*, *name=None*, *tex_name=None*, *info=None*, *cache=True*)

Methods

assign_memory(n)	Assign memory for `self.v` and set the array to zero.
get_names()	Return *name* in a list

Attributes

class_name	Return the class name
n	Return the count of values in `self.v`.
v	

andes.core.service.FlagValue

class andes.core.service.**FlagValue**(*u*, *value*, *flag=0*, *name=None*, *tex_name=None*, *info=None*, *cache=True*)

Class for flagging values that equal to the given value.

By default, values that equal to *value* will be flagged as *0*. Non-matching values will be flagged as *1*.

Parameters

u
Input parameter

value
Value to flag. Can be None, string, or a number.

flag
[0 by default, only 0 or 1 is accepted.] The flag for the matched ones

> **Warning:** *FlagNotNone* can only be applied to *BaseParam* with *cache=True*. Applying to *Service* will fail unless *cache* is False (at a performance cost).

__init__(*u*, *value*, *flag=0*, *name=None*, *tex_name=None*, *info=None*, *cache=True*)

Methods

assign_memory(n)	Assign memory for `self.v` and set the array to zero.
get_names()	Return *name* in a list

Attributes

class_name	Return the class name
n	Return the count of values in `self.v`.
v	

andes.core.service.FlagGreaterThan

class andes.core.service.**FlagGreaterThan**(*u*, *value=0.0*, *flag=1*, *equal=False*, *name=None*, *tex_name=None*, *info=None*, *cache=True*)

Service for flagging parameters > or >= the given value element-wise.

Parameters that satisfy the comparison (u > or >= value) will flagged as *flag* (1 by default).

__init__(*u*, *value=0.0*, *flag=1*, *equal=False*, *name=None*, *tex_name=None*, *info=None*, *cache=True*)

Methods

`assign_memory`(n)	Assign memory for `self.v` and set the array to zero.
`get_names`()	Return *name* in a list

Attributes

`class_name`	Return the class name
`n`	Return the count of values in `self.v`.
`v`	

andes.core.service.FlagLessThan

class andes.core.service.**FlagLessThan**(*u*, *value=0.0*, *flag=1*, *equal=False*, *name=None*, *tex_name=None*, *info=None*, *cache=True*)

Service for flagging parameters < or <= the given value element-wise.

Parameters that satisfy the comparison (u < or <= value) will flagged as *flag* (1 by default).

__init__(*u*, *value=0.0*, *flag=1*, *equal=False*, *name=None*, *tex_name=None*, *info=None*, *cache=True*)

Methods

`assign_memory`(n)	Assign memory for `self.v` and set the array to zero.
`get_names`()	Return *name* in a list

Attributes

`class_name`	Return the class name
`n`	Return the count of values in `self.v`.
`v`	

andes.core.service.InitChecker

class andes.core.service.**InitChecker**(*u*, *lower=None*, *upper=None*, *equal=None*, *not_equal=None*, *enable=True*, *error_out=False*, ***kwargs*)

Class for checking init values against known typical values.

Instances will be stored in *Model.services_post* and *Model.services_icheck*, which will be checked in *Model.post_init_check()* after initialization.

Parameters

u
v-provider to be checked

lower
[float, BaseParam, BaseVar, BaseService] lower bound

upper
[float, BaseParam, BaseVar, BaseService] upper bound

equal
[float, BaseParam, BaseVar, BaseService] values that the value from *v_str* should equal

not_equal
[float, BaseParam, BaseVar, BaseService] values that should not equal

enable
[bool] True to enable checking

Examples

Let's say generator excitation voltages are known to be in the range of 1.6 - 3.0 per unit. One can add the following instance to *GENBase*

```
self._vfc = InitChecker(u=self.vf,
                        info='vf range',
                        lower=1.8,
                        upper=3.0,
                        )
```

lower and *upper* can also take v-providers instead of float values.

One can also pass float values from Config to make it adjustable as in our implementation of `GENBase._vfc`.

__init__(*u*, *lower=None*, *upper=None*, *equal=None*, *not_equal=None*, *enable=True*, *error_out=False*, ***kwargs*)

Methods

`assign_memory`(n)	Assign memory for `self.v` and set the array to zero.
`check`()	Check the bounds and equality conditions.
`get_names`()	Return *name* in a list

Attributes

`class_name`	Return the class name
`n`	Return the count of values in `self.v`.
`v`	Return values stored in *self._v*.

andes.core.service.Replace

class andes.core.service.**Replace**(*old_val*, *flt*, *new_val*, *name=None*, *tex_name=None*, *info=None*, *cache=True*)

Replace parameters with new values if the function returns True

__init__(*old_val*, *flt*, *new_val*, *name=None*, *tex_name=None*, *info=None*, *cache=True*)

Methods

`assign_memory`(n)	Assign memory for `self.v` and set the array to zero.
`get_names`()	Return *name* in a list

Attributes

`class_name`	Return the class name
`n`	Return the count of values in `self.v`.
`v`	

andes.core.service.ApplyFunc

class `andes.core.service.`**`ApplyFunc`**(*u*, *func*, *name=None*, *tex_name=None*, *info=None*, *cache=True*)

Class for applying a numerical function on a parameter..

Parameters

u
Input parameter

func
A condition function that returns True or False.

> **Warning:** This class is not ready.

`__init__`(*u*, *func*, *name=None*, *tex_name=None*, *info=None*, *cache=True*)

Methods

`assign_memory`(n)	Assign memory for `self.v` and set the array to zero.
`get_names`()	Return *name* in a list

Attributes

`class_name`	Return the class name
`n`	Return the count of values in `self.v`.
`v`	

Class	Description
`ConstService`	Internal service for constant values
`VarService`	Variable service updated at each iteration before equations
`ExtService`	External service for retrieving values from value providers
`PostInitService`	Constant service evaluated after TDS initialization
`NumReduce`	Reduce linear 2-D arrays into 1-D arrays
`NumRepeat`	Repeat a 1-D array to linear 2-D arrays
`IdxRepeat`	Repeat a 1-D list to linear 2-D list
`EventFlag`	Flag changes in inputs as an event
`VarHold`	Hold input value when a hold signal is active
`ExtendedEvent`	Extend an event signal for a given period of time
`DataSelect`	Select optional str data if provided or use the fallback
`NumSelect`	Select optional numerical data if provided
`DeviceFinder`	Find or create devices linked to the given devices
`BackRef`	Collect idx-es for backward references
`RefFlatten`	Convert BackRef list of lists into a 1-D list
`InitChecker`	Check initial values against typical values
`FlagValue`	Flag values that equal the given value
`Replace`	Replace values that return True for the given lambda func

Internal Constants

The most commonly used service is `ConstService`. It stores an array of constants whose value is evaluated from a provided symbolic string. Constants are only evaluated once in the model initialization phase, ahead of variable initialization. `ConstService` is handy for calculating intermediate constants from parameters.

For example, a turbine governor has a `NumParam` `R` for the droop. `ConstService` allows calculating the inverse of the droop (the gain) and using it in equations:

```
self.R = NumParam()
self.G = ConstService(v_str='u/R')
```

where `u` is the online status parameter. The model can then use `G` in subsequent variable or equation strings.

class andes.core.service.**ConstService**(*v_str: str | None = None, v_numeric: Callable | None = None, vtype: type | None = None, name: str | None = None, tex_name: str | None = None, info: str | None = None, unit: str | None = None*)

A type of Service that stays constant once initialized.

ConstService are usually constants calculated from parameters. They are only evaluated once in the initialization phase before variables are initialized. Therefore, uninitialized variables must not be used in *v_str*`.

ConstService are evaluated *in sequence* after getting external variables and parameters and before initializing internal variables.

Parameters

name
[str] Name of the ConstService

v_str
[str] An equation string to calculate the variable value.

v_numeric
[Callable, optional] A callable which returns the value of the ConstService

v_type: type, optional, default to float
Type of element in the value array in float or complex

Attributes

v
[array-like or a scalar] ConstService value

_v_t0
[np.ndarray or None] `v` at t=0, saved by `snapshot_init()`. Used by `restore_init()` to reset `v` to the post-init baseline.

VarService

Updated at each iteration before equation evaluation. Use for intermediate values that depend on the current state of variables.

class `andes.core.service.`**`VarService`**(*v_str: str | None = None, v_numeric: Callable | None = None, vtype: type | None = None, name: str | None = None, tex_name: str | None = None, info: str | None = None, unit: str | None = None, sequential: bool | None = True*)

Variable service that gets updated in each step/iteration before computing the residual equations. As a results, variable values from the k-th step are used to compute a `VarService` that will be used to compute the residual for the (k+1)-th step.

This class is useful when one has non-differentiable algebraic equations, which make use of *abs()*, *re* and *im*. Instead of creating *Algeb*, one can put the equation in *VarService*, which will be updated before solving algebraic equations.

Parameters

sequential
[bool, optional, default to True] True if this VarService depends on previously defined VarService and should be evaluated in sequence. False if this VarService only uses known variables.

> **Warning:** `VarService` is intended for non-differentiable expressions (e.g., `Abs()`, `re()`, `im()`) that cannot appear in algebraic equations. If the expression is differentiable, an `Algeb` variable should be used instead.
>
> During model initialization, `VarService` is evaluated *before* any variable `v_str` assignment is computed. If the expression references algebraic or state variables, those variables will still hold their default (typically zero) values, resulting in an incorrect initial `VarService` value. Variables whose `v_str` depends on this `VarService` will consequently be initialized from an erroneous starting point. At runtime, this effect is self-correcting because `VarService` is re-evaluated every iteration with updated variable values. However, the incorrect initial value can cause convergence difficulties or incorrect limiter flag settings during the first few iterations.
>
> `VarService` is not solved simultaneously with algebraic equations, meaning that a one-step delay exists between the algebraic variables and the `VarService` value.

Examples

In ESST3A model, the voltage and current sensors (vd + jvq), (Id + jIq) estimate the sensed VE using equation

$$VE = |K_{PC} * (v_d + 1jv_q) + 1j(K_I + K_{PC} * X_L) * (I_d + 1jI_q)|$$

One can use *VarService* to implement this equation

```
self.VE = VarService(
    tex_name='V_E', info='VE', v_str='Abs(KPC*(vd + 1j*vq) + 1j*(KI +
    KPC*XL)*(Id + 1j*Iq))', )
```

PostInitService

Evaluated after TDS initialization, when variable values have been determined.

class andes.core.service.**PostInitService**(*v_str: str | None = None, v_numeric: Callable | None = None, vtype: type | None = None, name: str | None = None, tex_name: str | None = None, info: str | None = None, unit: str | None = None*)

Constant service that gets stored once after init.

This service is useful when one need to store initialization values stored in variables.

Examples

In ESST3A model, the *vf* variable is initialized followed by other variables. One can store the initial *vf* into *vf0* so that equation `vf - vf0 = 0` will hold.

```
self.vref0 = PostInitService(info='Initial reference voltage input',
                             tex_name='V_{ref0}', v_str='vref', )
```

Since all *ConstService* are evaluated before equation evaluation, without using PostInitService, one will need to create lots of *ConstService* to store values in the initialization path towards *vf0*, in order to correctly initialize *vf*.

External Constants

Service constants whose value is retrieved from an external model or group. Using `ExtService` is similar to using external variables. The values of `ExtService` will be retrieved once during the initialization phase before `ConstService` evaluation.

For example, a synchronous generator needs to retrieve the `p` and `q` values from static generators for initialization. In the `__init__()` of a synchronous generator model, one can define:

```
self.p0 = ExtService(src='p',
                     model='StaticGen',
                     indexer=self.gen,
                     tex_name='P_0')
```

class andes.core.service.**ExtService**(*model: str, src: str, indexer:* BaseParam | BaseService, *attr: str = 'v', allow_none: bool = False, default=0, name: str = None, tex_name: str = None, vtype=None, info: str = None*)

Service constants whose value is from an external model or group.

Parameters

src
[str] Variable or parameter name in the source model or group

model
[str] A model name or a group name

indexer
[IdxParam or BaseParam] An "Indexer" instance whose `v` field contains the `idx` of devices in the model or group.

Examples

A synchronous generator needs to retrieve the `p` and `q` values from static generators for initialization. `ExtService` is used for this purpose.

In a synchronous generator, one can define the following to retrieve `StaticGen.p` as `p0`:

```
class GENCLSModel(Model):
    def __init__(...):
        ...
        self.p0 = ExtService(src='p',
                             model='StaticGen',
                             indexer=self.gen,
                             tex_name='P_0')
```

Shape Manipulators

This section is for advanced model developers.

All generated equations operate on 1-dimensional arrays and can use algebraic calculations only. In some cases, one model would use `BackRef` to retrieve 2-dimensional indices and use such indices to retrieve variable addresses. The retrieved addresses usually have a different length than the referencing model and cannot be used directly for calculation.

Shape manipulator services can be used in such cases:

- `NumReduce` reduces a linearly stored 2-D ExtParam into 1-D Service
- `NumRepeat` repeats a 1-D value into linearly stored 2-D value based on the shape from a `BackRef`

class `andes.core.service.`**`BackRef`**(***kwargs*)

A special type of reference collector.

BackRef is used for collecting device indices of other models referencing the parent model of the *BackRef*. The *v``field will be a list of lists, each containing the `idx* of other models referencing each device of the parent model.

BackRef can be passed as indexer for params and vars, or shape for *NumReduce* and *NumRepeat*. See examples for illustration.

See also:

andes.core.service.NumReduce

A more complete example using BackRef to build the COI model

Examples

A Bus device has an *IdxParam* of *area*, storing the *idx* of area to which the bus device belongs. In `Bus.__init__()`, one has

```
self.area = IdxParam(model='Area')
```

Suppose *Bus* has the following data

idx	area	Vn
1	1	110
2	2	220
3	1	345
4	1	500

The Area model wants to collect the indices of Bus devices which points to the corresponding Area device. In `Area.__init__`, one defines

```
self.Bus = BackRef()
```

where the member attribute name *Bus* needs to match exactly model name that *Area* wants to collect *idx* for. Similarly, one can define `self.ACNode = BackRef()` to collect devices in the *ACNode* group that references Area.

The collection of *idx* happens in `andes.system.System._collect_ref_param()`. It has to be noted that the specific *Area* entry must exist to collect model idx-dx referencing it. For example, if *Area* has the following data

```
idx 1
```

Then, only Bus 1, 3, and 4 will be collected into *self.Bus.v*, namely, `self.Bus.v == [ [1, 3, 4] ]`.

If *Area* has data

```
idx 1 2
```

Then, *self.Bus.v* will end up with `[ [1, 3, 4], [2] ]`.

class `andes.core.service.`**`NumReduce`**(*u*, *ref:* BackRef, *fun:* *Callable*, *name=None*, *tex_name=None*, *info=None*, *cache=True*)

A helper Service type which reduces a linearly stored 2-D ExtParam into 1-D Service.

NumReduce works with ExtParam whose *v* field is a list of lists. A reduce function which takes an array-like and returns a scalar need to be supplied. NumReduce calls the reduce function on each of the lists and return all the scalars in an array.

Parameters

u

[ExtParam] Input ExtParam whose `v` contains linearly stored 2-dimensional values

ref
[BackRef] The BackRef whose 2-dimensional shapes are used for indexing

fun
[Callable] The callable for converting a 1-D array-like to a scalar

Examples

Suppose one wants to calculate the mean value of the `Vn` in one Area. In the `Area` class, one defines

```
class AreaModel(...):
    def __init__(...):
        ...
        # backward reference from `Bus`
        self.Bus = BackRef()

        # collect the Vn in an 1-D array
        self.Vn = ExtParam(model='Bus',
                           src='Vn',
                           indexer=self.Bus)

        self.Vn_mean = NumReduce(u=self.Vn,
                                 fun=np.mean,
                                 ref=self.Bus)
```

Suppose we define two areas, 1 and 2, the Bus data looks like

idx	area	Vn
1	1	110
2	2	220
3	1	345
4	1	500

Then, *self.Bus.v* is a list of two lists `[ [1, 3, 4], [2] ]`. *self.Vn.v* will be retrieved and linearly stored as `[110, 345, 500, 220]`. Based on the shape from *self.Bus*, `numpy.mean()` will be called on `[110, 345, 500]` and `[220]` respectively. Thus, *self.Vn_mean.v* will become `[318.33, 220]`.

class `andes.core.service.`**`NumRepeat`**(*u*, *ref*, ***kwargs*)

A helper Service type which repeats a v-provider's value based on the shape from a BackRef

Examples

NumRepeat was originally designed for computing the inertia-weighted average rotor speed (center of inertia speed). COI speed is computed with

$$\omega_{COI} = \frac{\sum M_i * \omega_i}{\sum M_i}$$

The numerator can be calculated with a mix of BackRef, ExtParam and ExtState. The denominator needs to be calculated with NumReduce and Service Repeat. That is, use NumReduce to calculate the sum, and use NumRepeat to repeat the summed value for each device.

In the COI class, one would have

```
class COIModel(...):
    def __init__(...):
        ...
        self.SynGen = BackRef()
        self.SynGenIdx = RefFlatten(ref=self.SynGen)
        self.M = ExtParam(model='SynGen',
                          src='M',
                          indexer=self.SynGenIdx)

        self.wgen = ExtState(model='SynGen',
                             src='omega',
                             indexer=self.SynGenIdx)

        self.Mt = NumReduce(u=self.M,
                                fun=np.sum,
                                ref=self.SynGen)

        self.Mtr = NumRepeat(u=self.Mt,
                               ref=self.SynGen)

        self.pidx = IdxRepeat(u=self.idx,ref=self.SynGen)
```

Finally, one would define the center of inertia speed as

```
self.wcoi = Algeb(v_str='1', e_str='-wcoi')

self.wcoi_sub = ExtAlgeb(model='COI',
                         src='wcoi',
                         e_str='M * wgen / Mtr',
                         v_str='M / Mtr',
                         indexer=self.pidx,
                         )
```

It is very worth noting that the implementation uses a trick to separate the average weighted sum into n sub-equations, each calculating the $(M_i * \omega_i)/(\sum M_i)$. Since all the variables are preserved in the sub-equation, the derivatives can be calculated correctly.

class andes.core.service.**IdxRepeat**(*u*, *ref*, ***kwargs*)

Helper class to repeat IdxParam.

This class has the same functionality as `andes.core.service.NumRepeat` but only operates on IdxParam, DataParam or NumParam.

class `andes.core.service.`**RefFlatten**(*ref*, ***kwargs*)

A service type for flattening `andes.core.service.BackRef` into a 1-D list.

Examples

This class is used when one wants to pass *BackRef* values as indexer.

`andes.models.coi.COI` collects referencing `andes.models.group.SynGen` with

```
self.SynGen = BackRef(info='SynGen idx lists', export=False)
```

After collecting BackRefs, *self.SynGen.v* will become a two-level list of indices, where the first level correspond to each COI and the second level correspond to generators of the COI.

Convert *self.SynGen* into 1-d as *self.SynGenIdx*, which can be passed as indexer for retrieving other parameters and variables

```
self.SynGenIdx = RefFlatten(ref=self.SynGen)

self.M = ExtParam(model='SynGen', src='M',
                  indexer=self.SynGenIdx, export=False,
                  )
```

Value Manipulation

class `andes.core.service.`**Replace**(*old_val*, *flt*, *new_val*, *name=None*, *tex_name=None*, *info=None*, *cache=True*)

Replace parameters with new values if the function returns True

class `andes.core.service.`**ParamCalc**(*param1*, *param2*, *func*, *name=None*, *tex_name=None*, *info=None*, *cache=True*)

Parameter calculation service.

Useful to create parameters calculated instantly from existing ones.

class `andes.core.service.`**ApplyFunc**(*u*, *func*, *name=None*, *tex_name=None*, *info=None*, *cache=True*)

Class for applying a numerical function on a parameter..

Parameters

u
Input parameter

func
A condition function that returns True or False.

> **Warning:** This class is not ready.

Idx and References

class andes.core.service.**DeviceFinder**(*u*, *link*, *idx_name: str*, *default_model: str*, *auto_find: bool | None = True*, *auto_add: bool | None = True*, *name: str | None = None*, *tex_name: str | None = None*, *info: str | None = None*)

Service for finding `idx` of devices which are linked to the given devices.

The `auto_find` parameter controls if the device idx should be automatically looked up. The `auto_add` parameter controls if the device will be automatically added. The two parameters are not exclusive. One can skip finding the device but automatically adding it.

If `auto_find` is `True` and the `idx` is None, DeviceFinder will look up for the device. If not found and `auto_add` is `True`, DevFinder will then automatically add the devices. The `idx` of the devices that are found or added will be stored to the DeviceFinder instance, so that *DeviceFinder* can be used like any *IdxParam*.

Adding new devices are called at the beginning of `andes.system.System.setup()`.

Examples

The IEEEST stabilizer takes an optional parameter `busf` of the type *IdxParam* for specifying the connected bus frequency measurement device, which is needed for mode 6. To avoid reimplementing *BusFreq* within IEEEST, one can do

```
self.busfreq = DeviceFinder(self.busf,
                            link=self.buss, idx_name='bus',
                            default_model='BusFreq')
```

where `self.busf` is for the optional parameter for the `idx` of bus frequency estimation devices (e.g., *BusFreq*), `self.buss` is for the `idx` of buses that `self.busf` devices should measure, and `idx_name` is the name of the BusFreq parameter through which the indices of measured buses are given.

For each `None` or invalid values in `self.busf`, a *BusFreq* device will be created with its `bus` set to the corresponding value in `self.buss`. That is, `BusFreq.[idx_name].v = [link]`.

At the end, the *DeviceFinder* instance will contain the list of `BusFreq` that are are connected to *self.buss*, respectively.

In the case of any valid value in `self.busf`, that is, the value is an existing `BusFreq` device, *DeviceFinder* will return it as is without checking if the `BusFreq` device actually measures the bus specified by `self.buss`. It allows to use the measurement at a different location, but the user have to perform the data consistency check.

Events

class andes.core.service.**EventFlag**(*u*, *vtype: type | None = None*, *name: str | None = None*, *tex_name=None*, *info=None*)

Service to flag events when the input value changes. The typical input is a *v-provider* with binary values.

Implemented by providing *self.check(**kwargs)* as *v_numeric*. *EventFlag.v* stores the values of the input variable in the most recent iteration/step.

After the evaluation of *self.check()*, *self.v* will be updated.

class andes.core.service.**ExtendedEvent**(*u*, *t_ext: int | float | BaseParam | BaseService = 0.0*, *trig: str = 'rise'*, *enable=True*, *v_disabled=0*, *extend_only=False*, *vtype: type | None = None*, *name: str | None = None*, *tex_name=None*, *info=None*)

Service for indicating an event for an extended, predefined period of time following the event disappearance.

The triggering of an event, whether the rise or fall edge, is specified through *trig*. For example, if *trig = rise*, the change of the input from 0 to 1 will be considered as an input, whereas the subsequent change back to 0 will be considered as the event end.

ExtendedEvent.v stores the flags whether the extended time has completed. Outputs will become 1 once the event starts and return to 0 when the extended time ends.

Parameters

u
[v-provider] Triggering signal where the values are 0 or 1.

trig
[str in ("rise", "fall")] Triggering edge for the beginning of an event. *rise* by default.

enable
[bool or v-provider] If disabled, the output will be *v_disabled*

extend_only
[bool] Only output during the extended period, not the event period.

> **Warning:** The performance of this class needs to be optimized.

class andes.core.service.**VarHold**(*u*, *hold*, *vtype=None*, *name=None*, *tex_name=None*, *info=None*)

Service for holding the input when the hold signal is on.

Parameters

hold
[v-provider, binary] Hold signal array with length equal to the input. For elements that are 1, the corresponding inputs are held until the hold signal returns to 0.

Flags

class andes.core.service.**FlagCondition**(*u*, *func*, *flag=1*, *name=None*, *tex_name=None*, *info=None*, *cache=True*)

Class for flagging values based on a condition function.

By default, values whose condition function output equal that equal to True/1 will be flagged as *1*. *0* otherwise.

Parameters

u
Input parameter

func
A condition function that returns True or False.

flag
[1 by default, only 0 or 1 is accepted.] The flag for the inputs whose condition output is True.

> **Warning:** This class is not ready.
>
> *FlagCondition* can only be applied to *BaseParam* with *cache=True*. Applying to *Service* will fail unless *cache* is False (at a performance cost).

class andes.core.service.**FlagGreaterThan**(*u*, *value=0.0*, *flag=1*, *equal=False*, *name=None*, *tex_name=None*, *info=None*, *cache=True*)

Service for flagging parameters > or >= the given value element-wise.

Parameters that satisfy the comparison (u > or >= value) will flagged as *flag* (1 by default).

class andes.core.service.**FlagLessThan**(*u*, *value=0.0*, *flag=1*, *equal=False*, *name=None*, *tex_name=None*, *info=None*, *cache=True*)

Service for flagging parameters < or <= the given value element-wise.

Parameters that satisfy the comparison (u < or <= value) will flagged as *flag* (1 by default).

class andes.core.service.**FlagValue**(*u*, *value*, *flag=0*, *name=None*, *tex_name=None*, *info=None*, *cache=True*)

Class for flagging values that equal to the given value.

By default, values that equal to *value* will be flagged as *0*. Non-matching values will be flagged as *1*.

Parameters

u
Input parameter

value
Value to flag. Can be None, string, or a number.

flag
[0 by default, only 0 or 1 is accepted.] The flag for the matched ones

> **Warning:** *FlagNotNone* can only be applied to *BaseParam* with *cache=True*. Applying to *Service* will fail unless *cache* is False (at a performance cost).

Data Select

class andes.core.service.**DataSelect**(*optional*, *fallback*, *name: str | None = None*, *tex_name: str | None = None*, *info: str | None = None*)

Class for selecting values for optional DataParam or NumParam.

This service is a v-provider that uses optional DataParam when available. Otherwise, use the fallback value.

DataParam will be tested for *None*, and NumParam will be tested with `np.isnan()`.

Notes

An use case of DataSelect is remote bus. One can do

```
self.buss = DataSelect(option=self.busr, fallback=self.bus)
```

Then, pass `self.buss` instead of `self.bus` as indexer to retrieve voltages.

Another use case is to allow an optional turbine rating. One can do

```
self.Tn = NumParam(default=None)
self.Sg = ExtParam(...)
self.Sn = DataSelect(Tn, Sg)
```

class andes.core.service.**NumSelect**(*optional*, *fallback*, *name: str | None = None*, *tex_name: str | None = None*, *info: str | None = None*, *ignore_cond: ~typing.Callable | None = functools.partial(<ufunc 'equal'>, 0)*)

Class for selecting values for optional NumParam.

NumSelect works with internal and external parameters.

Any values equal to `np.nan` will always be ignored. If one needs to ignore values based on additional conditions, pass it through `ignore_cond`. For example, to ignore zero values, use `ignore_cond = partial(np.equal, 0)`.

Examples

One use case is to allow an optional turbine rating. One can do

```
self.Tn = NumParam(default=None) self.Sg = ExtParam(...) self.Sn =
DataSelect(Tn, Sg)
```

Miscellaneous

class andes.core.service.**InitChecker**(*u*, *lower=None*, *upper=None*, *equal=None*, *not_equal=None*, *enable=True*, *error_out=False*, ***kwargs*)

Class for checking init values against known typical values.

Instances will be stored in *Model.services_post* and *Model.services_icheck*, which will be checked in *Model.post_init_check()* after initialization.

Parameters

u
v-provider to be checked

lower
[float, BaseParam, BaseVar, BaseService] lower bound

upper
[float, BaseParam, BaseVar, BaseService] upper bound

equal
[float, BaseParam, BaseVar, BaseService] values that the value from *v_str* should equal

not_equal
[float, BaseParam, BaseVar, BaseService] values that should not equal

enable
[bool] True to enable checking

Examples

Let's say generator excitation voltages are known to be in the range of 1.6 - 3.0 per unit. One can add the following instance to *GENBase*

```
self._vfc = InitChecker(u=self.vf,
                        info='vf range',
                        lower=1.8,
                        upper=3.0,
                        )
```

lower and *upper* can also take v-providers instead of float values.

One can also pass float values from Config to make it adjustable as in our implementation of `GENBase._vfc`.

class `andes.core.service.`**`CurrentSign`**(*bus*, *bus1*, *bus2*, *name=None*, *tex_name=None*, *info=None*)

Service for computing the sign of the current flowing through a series device.

With a given line connecting *bus1* and *bus2*, one can compute the current flow using `(v1*exp(1j*a1) - v2*exp(1j*a2)) / (r + jx)` whose value is the outflow on *bus1*.

CurrentSign can be used to compute the sign to be multiplied depending on the observing bus. For each value in *bus*, the sign will be `+1` if it appears in *bus1* or `-1` otherwise.

```
bus1          bus2
 *------>>-----*
bus(+)        bus(-)
```

class `andes.core.service.`**`RandomService`**(*func=<bound method RandomState.rand of RandomState(MT19937)>*, ***kwargs*)

A service type for generating random numbers.

Parameters

name
[str] Name

func
[Callable] A callable for generating the random variable.

> **Warning:** The value will be randomized every time it is accessed. Do not use it if the value needs to be stable for each simulation step.

class `andes.core.service.`**`SwBlock`**(**, *init*, *ns*, *blocks*, *ext_sel=None*, *name=None*, *tex_name=None*, *info=None*)

Service type for switched shunt blocks.

Common Patterns

Per-Unit Conversion

```
self.Zbase = ConstService(v_str='Vn**2 / Sn')
self.R_pu = ConstService(v_str='R_ohm / Zbase')
```

Inverse Calculation

```
self.R = NumParam()
self.gain = ConstService(v_str='u / R')
```

Conditional Value

```
self.K = ConstService(v_str='K1 * (mode == 1) + K2 * (mode == 2)')
```

See Also

- *Atomic Types* - v-provider concept
- *Parameters* - Input parameters
- *Variables* - DAE variables
- *Discrete Components* - Discrete components

4.3.4 Discrete Components

The discrete component library contains a special type of block for modeling discontinuities in power system devices. Such discontinuities can be device-level physical constraints or algorithmic limits imposed on controllers.

Background

The base class for discrete components is `andes.core.discrete.Discrete`.

The uniqueness of discrete components is the way they work. Discrete components take inputs, criteria, and export a set of flags with component-defined meanings. These exported flags can be used in algebraic or differential equations to build piece-wise equations.

For example, `Limiter` takes a v-provider as input and two v-providers as the upper and lower bounds. It exports three flags: `zi` (within bound), `zl` (below lower bound), and `zu` (above upper bound).

Update Timing

It is important to note when flags are updated. Discrete subclasses can use three methods to check and update values and equations:

Method	When Called	Purpose
`check_var`	Before equation evaluation	Variable-based flags (Limiter)
`check_eq`	After equation update	Equation-based flags (AntiWindup)
`set_var`	After solving	State pegging (AntiWindup)

In the current implementation:

- `check_var` updates flags for variable-based discrete components (such as `Limiter`)
- `check_eq` updates flags for equation-involved discrete components (such as `AntiWindup`)
- `set_var` is currently only used by `AntiWindup` to store the pegged states

Discrete Types

`Discrete`([name, tex_name, info, no_warn, ...])	Base discrete class.
`Limiter`(u, lower, upper[, enable, name, ...])	Base limiter class.
`SortedLimiter`(u, lower, upper[, n_select, ...])	A limiter that sorts inputs based on the absolute or relative amount of limit violations.
`HardLimiter`(u, lower, upper[, enable, name, ...])	Hard limiter for algebraic or differential variable.
`RateLimiter`(u, lower, upper[, enable, ...])	Rate limiter for a differential variable.
`AntiWindup`(u, lower, upper[, enable, ...])	Anti-windup limiter.
`AntiWindupRate`(u, lower, upper, rate_lower, ...)	Anti-windup limiter with rate limits
`LessThan`(u, bound[, equal, enable, name, ...])	Less than (<) comparison function that tests if `u < bound`.
`Selector`(*args, fun[, tex_name, info])	Selection between two variables using the provided reduce function.
`Switcher`(u, options[, info, name, tex_name, ...])	Switcher based on an input parameter.
`DeadBand`(u, center, lower, upper[, enable, ...])	The basic deadband type.
`DeadBandRT`(u, center, lower, upper[, enable])	Deadband with flags for directions of return.
`Average`(u[, mode, delay, name, tex_name, info])	Compute the average value of a BaseVar over a period of time or a number of simulation steps.
`Delay`(u[, mode, delay, name, tex_name, info])	The delay class.
`Derivative`(u[, name, tex_name, info])	Compute the derivative of a variable using numerical differentiation.
`Sampling`(u[, interval, offset, name, ...])	Sample and hold
`ShuntAdjust`(*, v, lower, upper, bsw, gsw, dt, u)	Class for adjusting switchable shunts.

andes.core.discrete.Discrete

class andes.core.discrete.**Discrete**(*name=None*, *tex_name=None*, *info=None*, *no_warn=False*, *min_iter=2*, *err_tol=0.01*)

Base discrete class.

Discrete classes export flag arrays (usually boolean) .

__init__(*name=None*, *tex_name=None*, *info=None*, *no_warn=False*, *min_iter=2*, *err_tol=0.01*)

Methods

check_eq(**kwargs)	This function is called in l_check_eq after updating equations.
check_iter_err([niter, err])	Check if the minimum iteration or maximum error is reached so that this discrete block should be enabled.
check_var(*args, **kwargs)	This function is called in l_update_var before evaluating equations.
get_limit_report()	Return a list of dicts describing variables clamped at limits.
get_names()	Available symbols from this class
get_tex_names()	Return tex_names of exported flags.
get_values()	
list2array(n)	Allocate memory for the discrete flags specified in *self.export_flags*.
save_unconstrained()	Save the input variable's unconstrained value (call after init pass 1).

Attributes

```
class_name
```

andes.core.discrete.Limiter

class andes.core.discrete.**Limiter**(*u*, *lower*, *upper*, *enable=True*, *name: str = None*, *tex_name: str = None*, *info: str = None*, *min_iter: int = 2*, *err_tol: float = 0.01*, *allow_adjust: bool = True*, *no_lower=False*, *no_upper=False*, *sign_lower=1*, *sign_upper=1*, *equal=True*, *no_warn=False*, *zu=0.0*, *zl=0.0*, *zi=1.0*)

Base limiter class.

This class compares values and sets limit values. Exported flags are *zi*, *zl* and *zu*.

Parameters

u
[BaseVar] Input Variable instance

lower
[BaseParam] Parameter instance for the lower limit

upper
[BaseParam] Parameter instance for the upper limit

no_lower
[bool] True to only use the upper limit

no_upper
[bool] True to only use the lower limit

sign_lower: 1 or -1
Sign to be multiplied to the lower limit

sign_upper: bool
Sign to be multiplied to the upper limit

equal
[bool] True to include equal signs in comparison (>= or <=).

no_warn
[bool] Disable initial limit warnings

zu
[0 or 1] Default value for *zu* if not enabled

zl
[0 or 1] Default value for *zl* if not enabled

zi
[0 or 1] Default value for *zi* if not enabled

Attributes

zl
[array-like] Flags of elements violating the lower limit; A array of zeros and/or ones.

zi
[array-like] Flags for within the limits

zu
[array-like] Flags for violating the upper limit

Notes

If not enabled, the default flags are `zu = zl = 0`, `zi = 1`.

__init__(*u*, *lower*, *upper*, *enable=True*, *name: str = None*, *tex_name: str = None*, *info: str = None*, *min_iter: int = 2*, *err_tol: float = 0.01*, *allow_adjust: bool = True*, *no_lower=False*, *no_upper=False*, *sign_lower=1*, *sign_upper=1*, *equal=True*, *no_warn=False*, *zu=0.0*, *zl=0.0*, *zi=1.0*)

Methods

`check_eq`(**kwargs)	This function is called in `l_check_eq` after updating equations.
`check_iter_err`([niter, err])	Check if the minimum iteration or maximum error is reached so that this discrete block should be enabled.
`check_var`([allow_adjust, adjust_lower, ...])	Check the input variable and set flags.
`do_adjust_lower`(val, lower[, allow_adjust, ...])	Adjust the lower limit.
`do_adjust_upper`(val, upper[, allow_adjust, ...])	Adjust the upper limit.
`get_limit_report`()	Return a list of dicts describing variables clamped at limits.
`get_names`()	Available symbols from this class
`get_tex_names`()	Return tex_names of exported flags.
`get_values`()	
`list2array`(n)	Allocate memory for the discrete flags specified in *self.export_flags*.
`save_unconstrained`()	Save the input variable's unconstrained value (call after init pass 1).

Attributes

```
class_name
```

andes.core.discrete.SortedLimiter

class andes.core.discrete.**SortedLimiter**(*u*, *lower*, *upper*, *n_select: int = 5*, *name=None*, *tex_name=None*, *enable=True*, *abs_violation=True*, *min_iter: int = 2*, *err_tol: float = 0.01*, *allow_adjust: bool = True*, *zu=0.0*, *zl=0.0*, *zi=1.0*, *ql=0.0*, *qu=0.0*)

A limiter that sorts inputs based on the absolute or relative amount of limit violations.

Parameters

n_select
[int] the number of violations to be flagged, for each of over-limit and under-limit cases. If *n_select* == 1, at most one over-limit and one under-limit inputs will be flagged. If *n_select* is zero, heuristics will be used.

abs_violation
[bool] True to use the absolute violation. False if the relative violation abs(violation/limit) is used for sorting. Since most variables are in per unit, absolute violation is recommended.

__init__(*u*, *lower*, *upper*, *n_select: int = 5*, *name=None*, *tex_name=None*, *enable=True*, *abs_violation=True*, *min_iter: int = 2*, *err_tol: float = 0.01*, *allow_adjust: bool = True*, *zu=0.0*, *zl=0.0*, *zi=1.0*, *ql=0.0*, *qu=0.0*)

Methods

`calc_select()`	Set *n_select* automatically.
`check_eq`(**kwargs)	This function is called in `l_check_eq` after updating equations.
`check_iter_err`([niter, err])	Check if the minimum iteration or maximum error is reached so that this discrete block should be enabled.
`check_var`(*args[, niter, err])	Check for the largest and smallest *n_select* elements.
`do_adjust_lower`(val, lower[, allow_adjust, ...])	Adjust the lower limit.
`do_adjust_upper`(val, upper[, allow_adjust, ...])	Adjust the upper limit.
`get_limit_report()`	Return a list of dicts describing variables clamped at limits.
`get_names()`	Available symbols from this class
`get_tex_names()`	Return tex_names of exported flags.
`get_values()`	
`list2array`(n)	Initialize maximum and minimum *n_select* based on input size.
`save_unconstrained()`	Save the input variable's unconstrained value (call after init pass 1).

Attributes

`class_name`

andes.core.discrete.HardLimiter

class andes.core.discrete.**HardLimiter**(*u*, *lower*, *upper*, *enable=True*, *name: str = None*, *tex_name: str = None*, *info: str = None*, *min_iter: int = 2*, *err_tol: float = 0.01*, *allow_adjust: bool = True*, *no_lower=False*, *no_upper=False*, *sign_lower=1*, *sign_upper=1*, *equal=True*, *no_warn=False*, *zu=0.0*, *zl=0.0*, *zi=1.0*)

Hard limiter for algebraic or differential variable. This class is an alias of *Limiter*.

__init__(*u*, *lower*, *upper*, *enable=True*, *name: str = None*, *tex_name: str = None*, *info: str = None*, *min_iter: int = 2*, *err_tol: float = 0.01*, *allow_adjust: bool = True*, *no_lower=False*, *no_upper=False*, *sign_lower=1*, *sign_upper=1*, *equal=True*, *no_warn=False*, *zu=0.0*, *zl=0.0*, *zi=1.0*)

Methods

`check_eq`(**kwargs)	This function is called in `l_check_eq` after updating equations.
`check_iter_err`([niter, err])	Check if the minimum iteration or maximum error is reached so that this discrete block should be enabled.
`check_var`([allow_adjust, adjust_lower, ...])	Check the input variable and set flags.
`do_adjust_lower`(val, lower[, allow_adjust, ...])	Adjust the lower limit.
`do_adjust_upper`(val, upper[, allow_adjust, ...])	Adjust the upper limit.
`get_limit_report`()	Return a list of dicts describing variables clamped at limits.
`get_names`()	Available symbols from this class
`get_tex_names`()	Return tex_names of exported flags.
`get_values`()	
`list2array`(n)	Allocate memory for the discrete flags specified in *self.export_flags*.
`save_unconstrained`()	Save the input variable's unconstrained value (call after init pass 1).

Attributes

`class_name`

andes.core.discrete.RateLimiter

class `andes.core.discrete.`**`RateLimiter`**(*u*, *lower*, *upper*, *enable=True*, *no_lower=False*, *no_upper=False*, *lower_cond=1*, *upper_cond=1*, *name=None*, *tex_name=None*, *info=None*)

Rate limiter for a differential variable.

RateLimiter does not export any variable. It directly modifies the differential equation value.

> **Warning:** RateLimiter cannot be applied to a state variable that already undergoes an AntiWindup limiter. Use *AntiWindupRate* for a rate-limited anti-windup limiter.

Notes

RateLimiter inherits from Discrete to avoid internal naming conflicts with *Limiter*.

__init__(*u*, *lower*, *upper*, *enable=True*, *no_lower=False*, *no_upper=False*, *lower_cond=1*, *upper_cond=1*, *name=None*, *tex_name=None*, *info=None*)

Methods

`check_eq`(**kwargs)	This function is called in `l_check_eq` after updating equations.
`check_iter_err`([niter, err])	Check if the minimum iteration or maximum error is reached so that this discrete block should be enabled.
`check_var`(*args, **kwargs)	This function is called in `l_update_var` before evaluating equations.
`get_limit_report`()	Return a list of dicts describing variables clamped at limits.
`get_names`()	Available symbols from this class
`get_tex_names`()	Return tex_names of exported flags.
`get_values`()	
`list2array`(n)	Allocate memory for the discrete flags specified in *self.export_flags*.
`save_unconstrained`()	Save the input variable's unconstrained value (call after init pass 1).

Attributes

`class_name`

andes.core.discrete.AntiWindup

class andes.core.discrete.**AntiWindup**(*u*, *lower*, *upper*, *enable=True*, *no_warn=False*, *no_lower=False*, *no_upper=False*, *sign_lower=1*, *sign_upper=1*, *name=None*, *tex_name=None*, *info=None*, *state=None*, *allow_adjust: bool = True*)

Anti-windup limiter.

Anti-windup limiter prevents the wind-up effect of a differential variable. The derivative of the differential variable is reset if it continues to increase in the same direction after exceeding the limits. During the derivative return, the limiter will be inactive

```
if x > xmax and x dot > 0: x = xmax and x dot = 0
if x < xmin and x dot < 0: x = xmin and x dot = 0
```

This class takes one more optional parameter for specifying the equation.

Parameters

state
[State, ExtState] A State (or ExtState) whose equation value will be checked and, when condition satisfies, will be reset by the anti-windup-limiter.

__init__(*u*, *lower*, *upper*, *enable=True*, *no_warn=False*, *no_lower=False*, *no_upper=False*, *sign_lower=1*, *sign_upper=1*, *name=None*, *tex_name=None*, *info=None*, *state=None*, *allow_adjust: bool = True*)

Methods

check_eq([allow_adjust, adjust_lower, ...])	Check the variables and equations and set the limiter flags.
check_iter_err([niter, err])	Check if the minimum iteration or maximum error is reached so that this discrete block should be enabled.
check_var(*args, **kwargs)	This function is empty.
do_adjust_lower(val, lower[, allow_adjust, ...])	Adjust the lower limit.
do_adjust_upper(val, upper[, allow_adjust, ...])	Adjust the upper limit.
get_limit_report()	Return a list of dicts describing variables clamped at limits.
get_names()	Available symbols from this class
get_tex_names()	Return tex_names of exported flags.
get_values()	
list2array(n)	Allocate memory for the discrete flags specified in *self.export_flags*.
save_unconstrained()	Save the input variable's unconstrained value (call after init pass 1).

Attributes

class_name

andes.core.discrete.AntiWindupRate

class andes.core.discrete.**AntiWindupRate**(*u*, *lower*, *upper*, *rate_lower*, *rate_upper*, *no_lower=False*, *no_upper=False*, *rate_no_lower=False*, *rate_no_upper=False*, *rate_lower_cond=None*, *rate_upper_cond=None*, *enable=True*, *name=None*, *tex_name=None*, *info=None*, *allow_adjust: bool = True*)

Anti-windup limiter with rate limits

__init__(*u*, *lower*, *upper*, *rate_lower*, *rate_upper*, *no_lower=False*, *no_upper=False*, *rate_no_lower=False*, *rate_no_upper=False*, *rate_lower_cond=None*, *rate_upper_cond=None*, *enable=True*, *name=None*, *tex_name=None*, *info=None*, *allow_adjust: bool = True*)

Methods

check_eq(**kwargs)	Check the variables and equations and set the limiter flags.
check_iter_err([niter, err])	Check if the minimum iteration or maximum error is reached so that this discrete block should be enabled.
check_var(*args, **kwargs)	This function is empty.
do_adjust_lower(val, lower[, allow_adjust, ...])	Adjust the lower limit.
do_adjust_upper(val, upper[, allow_adjust, ...])	Adjust the upper limit.
get_limit_report()	Return a list of dicts describing variables clamped at limits.
get_names()	Available symbols from this class
get_tex_names()	Return tex_names of exported flags.
get_values()	
list2array(n)	Allocate memory for the discrete flags specified in *self.export_flags*.
save_unconstrained()	Save the input variable's unconstrained value (call after init pass 1).

Attributes

class_name

andes.core.discrete.LessThan

class andes.core.discrete.**LessThan**(*u*, *bound*, *equal=False*, *enable=True*, *name=None*, *tex_name=None*, *info: str = None*, *cache: bool = False*, *z0=0*, *z1=1*)

Less than (<) comparison function that tests if `u < bound`.

Exports two flags: z1 and z0. For elements satisfying the less-than condition, the corresponding z1 = 1. z0 is the element-wise negation of z1.

Notes

The default z0 and z1, if not enabled, can be set through the constructor. By default, the model will not adjust the limit.

__init__(*u*, *bound*, *equal=False*, *enable=True*, *name=None*, *tex_name=None*, *info: str = None*, *cache: bool = False*, *z0=0*, *z1=1*)

Methods

check_eq(**kwargs)	This function is called in `l_check_eq` after updating equations.
check_iter_err([niter, err])	Check if the minimum iteration or maximum error is reached so that this discrete block should be enabled.
check_var(*args, **kwargs)	If enabled, set flags based on inputs.
get_limit_report()	Return a list of dicts describing variables clamped at limits.
get_names()	Available symbols from this class
get_tex_names()	Return tex_names of exported flags.
get_values()	
list2array(n)	Allocate memory for the discrete flags specified in *self.export_flags*.
save_unconstrained()	Save the input variable's unconstrained value (call after init pass 1).

Attributes

```
class_name
```

andes.core.discrete.Selector

class andes.core.discrete.**Selector**(**args*, *fun*, *tex_name=None*, *info=None*)

Selection between two variables using the provided reduce function.

The reduce function should take the given number of arguments. An example function is *np.maximum.reduce* which can be used to select the maximum.

Names are in *s0*, *s1*.

> **Warning:** A potential bug when more than two inputs are provided, and values in different inputs are equal. Only two inputs are allowed.
>
> Deprecated since version 1.5.9: Use of this class for comparison-based output is discouraged. Instead, use *LessThan* and *Limiter* to construct piesewise equations.
>
> See the new implementation of `HVGate` and `LVGate`.

See also:

numpy.ufunc.reduce
NumPy reduce function

Notes

A common pitfall is the 0-based indexing in the Selector flags. Note that exported flags start from 0. Namely, *s0* corresponds to the first variable provided for the Selector constructor.

Examples

Example 1: select the largest value between *v0* and *v1* and put it into vmax.

After the definitions of *v0* and *v1*, define the algebraic variable *vmax* for the largest value, and a selector *vs*

```
self.vmax = Algeb(v_str='maximum(v0, v1)',
                  tex_name='v_{max}',
                  e_str='vs_s0 * v0 + vs_s1 * v1 - vmax')

self.vs = Selector(self.v0, self.v1, fun=np.maximum.reduce)
```

The initial value of *vmax* is calculated by `maximum(v0, v1)`, which is the element-wise maximum in SymPy and will be generated into `np.maximum(v0, v1)`. The equation of *vmax* is to select the values based on *vs_s0* and *vs_s1*.

__init__(**args*, *fun*, *tex_name=None*, *info=None*)

Methods

`check_eq`(**kwargs)	This function is called in `l_check_eq` after updating equations.
`check_iter_err`([niter, err])	Check if the minimum iteration or maximum error is reached so that this discrete block should be enabled.
`check_var`(*args, **kwargs)	Set the i-th variable's flags to 1 if the return of the reduce function equals the i-th input.
`get_limit_report`()	Return a list of dicts describing variables clamped at limits.
`get_names`()	Available symbols from this class
`get_tex_names`()	Return tex_names of exported flags.
`get_values`()	
`list2array`(n)	Allocate memory for the discrete flags specified in *self.export_flags*.
`save_unconstrained`()	Save the input variable's unconstrained value (call after init pass 1).

Attributes

`class_name`

andes.core.discrete.Switcher

class `andes.core.discrete.`**Switcher**(*u*, *options: list | tuple*, *info: str = None*, *name: str = None*, *tex_name: str = None*, *cache=True*)

Switcher based on an input parameter.

The switch class takes one v-provider, compares the input with each value in the option list, and exports one flag array for each option. The flags are 0-indexed.

Exported flags are named with *_s0*, *_s1*, ..., with a total number of *len(options)*. See the examples section.

Notes

Switches needs to be distinguished from Selector.

Switcher is for generating flags indicating option selection based on an input parameter. Selector is for generating flags at run time based on variable values and a selection function.

Examples

The IEEEST model takes an input for selecting the signal. Options are 1 through 6. One can construct

```
self.IC = NumParam(info='input code 1-6')  # input code
self.SW = Switcher(u=self.IC, options=[0, 1, 2, 3, 4, 5, 6])
```

If the IC values from the data file ends up being

```
self.IC.v = np.array([1, 2, 2, 4, 6])
```

Then, the exported flag arrays will be

```
{'IC_s0': np.array([0, 0, 0, 0, 0]),
 'IC_s1': np.array([1, 0, 0, 0, 0]),
 'IC_s2': np.array([0, 1, 1, 0, 0]),
 'IC_s3': np.array([0, 0, 0, 0, 0]),
 'IC_s4': np.array([0, 0, 0, 1, 0]),
 'IC_s5': np.array([0, 0, 0, 0, 0]),
 'IC_s6': np.array([0, 0, 0, 0, 1])
}
```

where *IC_s0* is used for padding so that following flags align with the options.

__init__(*u*, *options: list | tuple*, *info: str = None*, *name: str = None*, *tex_name: str = None*, *cache=True*)

Methods

check_eq(**kwargs)	This function is called in l_check_eq after updating equations.
check_iter_err([niter, err])	Check if the minimum iteration or maximum error is reached so that this discrete block should be enabled.
check_var(*args, **kwargs)	Set the switcher flags based on inputs.
get_limit_report()	Return a list of dicts describing variables clamped at limits.
get_names()	Available symbols from this class
get_tex_names()	Return tex_names of exported flags.
get_values()	
list2array(n)	This forces to evaluate Switcher upon System setup
save_unconstrained()	Save the input variable's unconstrained value (call after init pass 1).

Attributes

class_name

andes.core.discrete.DeadBand

class andes.core.discrete.**DeadBand**(*u*, *center*, *lower*, *upper*, *enable=True*, *equal=False*, *zu=0.0*, *zl=0.0*, *zi=1.0*, *name=None*, *tex_name=None*, *info=None*)

The basic deadband type.

Parameters

u
[NumParam] The pre-deadband input variable

center
[NumParam] Neutral value of the output

lower
[NumParam] Lower bound

upper
[NumParam] Upper bound

enable
[bool] Enabled if True; Disabled and works as a pass-through if False.

Notes

Input changes within a deadband will incur no output changes. This component computes and exports three flags.

Three flags computed from the current input:

- zl: True if the input is below the lower threshold
- zi: True if the input is within the deadband
- zu: True if is above the lower threshold

Initial condition:

All three flags are initialized to zero. All flags are updated during *check_var* when enabled. If the deadband component is not enabled, all of them will remain zero.

Examples

Exported deadband flags need to be used in the algebraic equation corresponding to the post-deadband variable. Assume the pre-deadband input variable is *var_in* and the post-deadband variable is *var_out*. First, define a deadband instance *db* in the model using

```
self.db = DeadBand(u=self.var_in, center=self.dbc,
                   lower=self.dbl, upper=self.dbu)
```

To implement a no-memory deadband whose output returns to center when the input is within the band, the equation for *var* can be written as

```
var_out.e_str = 'var_in * (1 - db_zi) + \
                 (dbc * db_zi) - var_out'
```

__init__(*u*, *center*, *lower*, *upper*, *enable=True*, *equal=False*, *zu=0.0*, *zl=0.0*, *zi=1.0*, *name=None*, *tex_name=None*, *info=None*)

Methods

check_eq(**kwargs)	This function is called in l_check_eq after updating equations.
check_iter_err([niter, err])	Check if the minimum iteration or maximum error is reached so that this discrete block should be enabled.
check_var(*args, **kwargs)	**Notes**
do_adjust_lower(val, lower[, allow_adjust, ...])	Adjust the lower limit.
do_adjust_upper(val, upper[, allow_adjust, ...])	Adjust the upper limit.
get_limit_report()	Return a list of dicts describing variables clamped at limits.
get_names()	Available symbols from this class
get_tex_names()	Return tex_names of exported flags.
get_values()	
list2array(n)	Allocate memory for the discrete flags specified in *self.export_flags*.
save_unconstrained()	Save the input variable's unconstrained value (call after init pass 1).

Attributes

class_name

andes.core.discrete.DeadBandRT

class andes.core.discrete.**DeadBandRT**(*u*, *center*, *lower*, *upper*, *enable=True*)

Deadband with flags for directions of return.

Parameters

u
[NumParam] The pre-deadband input variable

center
[NumParam] Neutral value of the output

lower
[NumParam] Lower bound

upper
[NumParam] Upper bound

enable
[bool] Enabled if True; Disabled and works as a pass-through if False.

Notes

Input changes within a deadband will incur no output changes. This component computes and exports five flags. The additional two flags on top of *DeadBand* indicate the direction of return:

- zur: True if the input is/has been within the deadband and was returned from the upper threshold
- zlr: True if the input is/has been within the deadband and was returned from the lower threshold

Initial condition:

All five flags are initialized to zero. All flags are updated during *check_var* when enabled. If the deadband component is not enabled, all of them will remain zero.

Examples

To implement a deadband whose output is pegged at the nearest deadband bounds, the equation for *var* can be provided as

```
var_out.e_str = 'var_in * (1 - db_zi) + \
                 dbl * db_zlr + \
                 dbu * db_zur - var_out'
```

__init__(*u*, *center*, *lower*, *upper*, *enable=True*)

Initialize DeadBandRT with return-direction tracking flags.

Methods

`check_eq`(**kwargs)	This function is called in `l_check_eq` after updating equations.
`check_iter_err`([niter, err])	Check if the minimum iteration or maximum error is reached so that this discrete block should be enabled.
`check_var`(*args, **kwargs)	**Notes**
`do_adjust_lower`(val, lower[, allow_adjust, ...])	Adjust the lower limit.
`do_adjust_upper`(val, upper[, allow_adjust, ...])	Adjust the upper limit.
`get_limit_report`()	Return a list of dicts describing variables clamped at limits.
`get_names`()	Available symbols from this class
`get_tex_names`()	Return tex_names of exported flags.
`get_values`()	
`list2array`(n)	Allocate memory for the discrete flags specified in *self.export_flags*.
`save_unconstrained`()	Save the input variable's unconstrained value (call after init pass 1).

Attributes

`class_name`

andes.core.discrete.Average

class `andes.core.discrete.`**`Average`**(*u*, *mode='step'*, *delay=0*, *name=None*, *tex_name=None*, *info=None*)

Compute the average value of a BaseVar over a period of time or a number of simulation steps.

Average is based on the memory implemented in the Delay class. The same modes as in Delay are supported.

The output of the Average class is named `<INSTANCE_NAME>_v`, where *<INSTANCE_NAME>* is the instance name of Average.

`__init__`(*u*, *mode='step'*, *delay=0*, *name=None*, *tex_name=None*, *info=None*)

Methods

check_eq(**kwargs)	This function is called in l_check_eq after updating equations.
check_iter_err([niter, err])	Check if the minimum iteration or maximum error is reached so that this discrete block should be enabled.
check_var(dae_t, *args, **kwargs)	This function is called in l_update_var before evaluating equations.
get_limit_report()	Return a list of dicts describing variables clamped at limits.
get_names()	Available symbols from this class
get_tex_names()	Return tex_names of exported flags.
get_values()	
list2array(n)	Allocate memory for storage arrays.
save_unconstrained()	Save the input variable's unconstrained value (call after init pass 1).

Attributes

class_name

andes.core.discrete.Delay

class andes.core.discrete.**Delay**(*u*, *mode='step'*, *delay=0*, *name=None*, *tex_name=None*, *info=None*)

The delay class.

Delay allows to impose a predefined and fixed "delay" (in either steps or seconds) for an input variable. The amount of delay must be a scalar and has to be given when instantiating the Delay class when defining the model.

Delay implements an internal memorize to store past variable values.

The default delay mode is *step* but can be set to *time*. In the *time* mode, the value at the `current time - delay` will be interpolated based on the two nearest times and values.

Delay can be applied to a state or an algebraic variable. The exported variable is named `<INSTANCE_NAME>_v`, where *<INSTANCE_NAME>* is the name of the Delay instance.

__init__(*u*, *mode='step'*, *delay=0*, *name=None*, *tex_name=None*, *info=None*)

Methods

check_eq(**kwargs)	This function is called in l_check_eq after updating equations.
check_iter_err([niter, err])	Check if the minimum iteration or maximum error is reached so that this discrete block should be enabled.
check_var(dae_t, *args, **kwargs)	This function is called in l_update_var before evaluating equations.
get_limit_report()	Return a list of dicts describing variables clamped at limits.
get_names()	Available symbols from this class
get_tex_names()	Return tex_names of exported flags.
get_values()	
list2array(n)	Allocate memory for storage arrays.
save_unconstrained()	Save the input variable's unconstrained value (call after init pass 1).

Attributes

class_name

andes.core.discrete.Derivative

class andes.core.discrete.**Derivative**(*u*, *name=None*, *tex_name=None*, *info=None*)

Compute the derivative of a variable using numerical differentiation.

Derivative is based on the storage implemented in the Delay class. The delay is set to *1* step so that the current and the previous step are used.

A simple first order derivative is computed using `u(t) - u(t-1) / tstep`, where `tstep` is the current step size.

Derivative is intended to be used for algebraic variables because of discontinuity. It can be applied to a state variable, but one should instead implement the right-hand side equation of the state variable in an algebraic equation to obtain the accurate derivative.

Alternatively, the washout filter (*andes.core.block.Washout*) can be used to implement a numerically stable derivative.

The output of the Derivative class is named `<INSTANCE_NAME>_v` just like *Delay*.

__init__(*u*, *name=None*, *tex_name=None*, *info=None*)

Methods

`check_eq`(**kwargs)	This function is called in `l_check_eq` after updating equations.
`check_iter_err`([niter, err])	Check if the minimum iteration or maximum error is reached so that this discrete block should be enabled.
`check_var`(dae_t, *args, **kwargs)	Calculate the numerical differentiation.
`get_limit_report`()	Return a list of dicts describing variables clamped at limits.
`get_names`()	Available symbols from this class
`get_tex_names`()	Return tex_names of exported flags.
`get_values`()	
`list2array`(n)	Allocate memory for storage arrays.
`save_unconstrained`()	Save the input variable's unconstrained value (call after init pass 1).

Attributes

`class_name`

andes.core.discrete.Sampling

class `andes.core.discrete.`**`Sampling`**(*u*, *interval=1.0*, *offset=0.0*, *name=None*, *tex_name=None*, *info=None*)

Sample and hold

Sample an input variable periodically at the given time interval and hold the value until the next sample time.

For example, this class can be used to implement a 4-second sampling of the AGC signal.

The output of *Sampling* is named `<INSTANCE_NAME>_v`, where *<INSTANCE_NAME>* is the Sampling instance name.

`__init__`(*u*, *interval=1.0*, *offset=0.0*, *name=None*, *tex_name=None*, *info=None*)

Methods

check_eq(**kwargs)	This function is called in l_check_eq after updating equations.
check_iter_err([niter, err])	Check if the minimum iteration or maximum error is reached so that this discrete block should be enabled.
check_var(dae_t, *args, **kwargs)	Check and update the output.
get_limit_report()	Return a list of dicts describing variables clamped at limits.
get_names()	Available symbols from this class
get_tex_names()	Return tex_names of exported flags.
get_values()	
list2array(n)	Allocate memory and set all _last_v internal storage to zeros.
save_unconstrained()	Save the input variable's unconstrained value (call after init pass 1).

Attributes

class_name

andes.core.discrete.ShuntAdjust

class andes.core.discrete.**ShuntAdjust**(*, *v*, *lower*, *upper*, *bsw*, *gsw*, *dt*, *u*, *enable=True*, *min_iter=2*, *err_tol=0.01*, *name=None*, *tex_name=None*, *info=None*, *no_warn=False*)

Class for adjusting switchable shunts.

Parameters

v
[BaseVar] Voltage measurement

lower
[BaseParam] Lower voltage bound

upper
[BaseParam] Upper voltage bound

bsw
[SwBlock] SwBlock instance for susceptance

gsw
[SwBlock] SwBlock instance for conductance

dt
[NumParam] Delay time

u
[NumParam] Connection status

min_iter
[int] Minimum iteration number to enable shunt switching

err_tol
[float] Minimum iteration tolerance to enable switching

__init__(*, *v*, *lower*, *upper*, *bsw*, *gsw*, *dt*, *u*, *enable=True*, *min_iter=2*, *err_tol=0.01*, *name=None*, *tex_name=None*, *info=None*, *no_warn=False*)

Methods

`check_eq`(**kwargs)	This function is called in `l_check_eq` after updating equations.
`check_iter_err`([niter, err])	Check if the minimum iteration or maximum error is reached so that this discrete block should be enabled.
`check_var`(dae_t, *args[, niter, err])	Check voltage and perform shunt switching.
`get_limit_report`()	Return a list of dicts describing variables clamped at limits.
`get_names`()	Available symbols from this class
`get_tex_names`()	Return tex_names of exported flags.
`get_values`()	
`list2array`(n)	Allocate memory for the discrete flags specified in *self.export_flags*.
`save_unconstrained`()	Save the input variable's unconstrained value (call after init pass 1).

Attributes

`class_name`

Limiters

class andes.core.discrete.**Limiter**(*u*, *lower*, *upper*, *enable=True*, *name: str = None*, *tex_name: str = None*, *info: str = None*, *min_iter: int = 2*, *err_tol: float = 0.01*, *allow_adjust: bool = True*, *no_lower=False*, *no_upper=False*, *sign_lower=1*, *sign_upper=1*, *equal=True*, *no_warn=False*, *zu=0.0*, *zl=0.0*, *zi=1.0*)

Base limiter class.

This class compares values and sets limit values. Exported flags are *zi*, *zl* and *zu*.

Parameters

u
[BaseVar] Input Variable instance

lower
[BaseParam] Parameter instance for the lower limit

upper
[BaseParam] Parameter instance for the upper limit

no_lower
[bool] True to only use the upper limit

no_upper
[bool] True to only use the lower limit

sign_lower: 1 or -1
Sign to be multiplied to the lower limit

sign_upper: bool
Sign to be multiplied to the upper limit

equal
[bool] True to include equal signs in comparison (>= or <=).

no_warn
[bool] Disable initial limit warnings

zu
[0 or 1] Default value for *zu* if not enabled

zl
[0 or 1] Default value for *zl* if not enabled

zi
[0 or 1] Default value for *zi* if not enabled

Attributes

zl
[array-like] Flags of elements violating the lower limit; A array of zeros and/or ones.

zi
[array-like] Flags for within the limits

zu
[array-like] Flags for violating the upper limit

Notes

If not enabled, the default flags are `zu = zl = 0`, `zi = 1`.

Example usage:

```
from andes.core.discrete import Limiter

self.lim = Limiter(
    u=self.x,         # Input variable
    lower=self.Vmin,  # Lower bound
    upper=self.Vmax   # Upper bound
)

# Flags exported:
# lim.zi = 1 when lower <= x <= upper (within)
# lim.zl = 1 when x < lower (below)
# lim.zu = 1 when x > upper (above)

# Use in equation for piecewise behavior
self.y = Algeb(e_str='x*lim_zi + upper*lim_zu + lower*lim_zl - y')
```

See the code in `models/static/pq.py` for an example of voltage-based PQ-to-Z conversion.

class andes.core.discrete.**SortedLimiter**(*u*, *lower*, *upper*, *n_select: int = 5*, *name=None*, *tex_name=None*, *enable=True*, *abs_violation=True*, *min_iter: int = 2*, *err_tol: float = 0.01*, *allow_adjust: bool = True*, *zu=0.0*, *zl=0.0*, *zi=1.0*, *ql=0.0*, *qu=0.0*)

A limiter that sorts inputs based on the absolute or relative amount of limit violations.

Parameters

n_select
[int] the number of violations to be flagged, for each of over-limit and under-limit cases. If *n_select* == 1, at most one over-limit and one under-limit inputs will be flagged. If *n_select* is zero, heuristics will be used.

abs_violation
[bool] True to use the absolute violation. False if the relative violation abs(violation/limit) is used for sorting. Since most variables are in per unit, absolute violation is recommended.

class andes.core.discrete.**HardLimiter**(*u*, *lower*, *upper*, *enable=True*, *name: str = None*, *tex_name: str = None*, *info: str = None*, *min_iter: int = 2*, *err_tol: float = 0.01*, *allow_adjust: bool = True*, *no_lower=False*, *no_upper=False*, *sign_lower=1*, *sign_upper=1*, *equal=True*, *no_warn=False*, *zu=0.0*, *zl=0.0*, *zi=1.0*)

Hard limiter for algebraic or differential variable. This class is an alias of *Limiter*.

class andes.core.discrete.**RateLimiter**(*u*, *lower*, *upper*, *enable=True*, *no_lower=False*, *no_upper=False*, *lower_cond=1*, *upper_cond=1*, *name=None*, *tex_name=None*, *info=None*)

Rate limiter for a differential variable.

RateLimiter does not export any variable. It directly modifies the differential equation value.

> **Warning:** RateLimiter cannot be applied to a state variable that already undergoes an AntiWindup limiter. Use *AntiWindupRate* for a rate-limited anti-windup limiter.

Notes

RateLimiter inherits from Discrete to avoid internal naming conflicts with *Limiter*.

class andes.core.discrete.**AntiWindup**(*u*, *lower*, *upper*, *enable=True*, *no_warn=False*, *no_lower=False*, *no_upper=False*, *sign_lower=1*, *sign_upper=1*, *name=None*, *tex_name=None*, *info=None*, *state=None*, *allow_adjust: bool = True*)

Anti-windup limiter.

Anti-windup limiter prevents the wind-up effect of a differential variable. The derivative of the differential variable is reset if it continues to increase in the same direction after exceeding the limits. During the derivative return, the limiter will be inactive

```
if x > xmax and x dot > 0: x = xmax and x dot = 0
if x < xmin and x dot < 0: x = xmin and x dot = 0
```

This class takes one more optional parameter for specifying the equation.

Parameters

state
[State, ExtState] A State (or ExtState) whose equation value will be checked and, when condition satisfies, will be reset by the anti-windup-limiter.

class andes.core.discrete.**AntiWindupRate**(*u*, *lower*, *upper*, *rate_lower*, *rate_upper*, *no_lower=False*, *no_upper=False*, *rate_no_lower=False*, *rate_no_upper=False*, *rate_lower_cond=None*, *rate_upper_cond=None*, *enable=True*, *name=None*, *tex_name=None*, *info=None*, *allow_adjust: bool = True*)

Anti-windup limiter with rate limits

Comparers

class `andes.core.discrete.`**`LessThan`**(*u*, *bound*, *equal=False*, *enable=True*, *name=None*, *tex_name=None*, *info: str = None*, *cache: bool = False*, *z0=0*, *z1=1*)

Less than (<) comparison function that tests if `u < bound`.

Exports two flags: z1 and z0. For elements satisfying the less-than condition, the corresponding z1 = 1. z0 is the element-wise negation of z1.

Notes

The default z0 and z1, if not enabled, can be set through the constructor. By default, the model will not adjust the limit.

class `andes.core.discrete.`**`Selector`**(**args*, *fun*, *tex_name=None*, *info=None*)

Selection between two variables using the provided reduce function.

The reduce function should take the given number of arguments. An example function is *np.maximum.reduce* which can be used to select the maximum.

Names are in *s0*, *s1*.

> **Warning:** A potential bug when more than two inputs are provided, and values in different inputs are equal. Only two inputs are allowed.
>
> Deprecated since version 1.5.9: Use of this class for comparison-based output is discouraged. Instead, use *LessThan* and *Limiter* to construct piesewise equations.
>
> See the new implementation of `HVGate` and `LVGate`.

See also:

`numpy.ufunc.reduce`
NumPy reduce function

Notes

A common pitfall is the 0-based indexing in the Selector flags. Note that exported flags start from 0. Namely, *s0* corresponds to the first variable provided for the Selector constructor.

Examples

Example 1: select the largest value between *v0* and *v1* and put it into vmax.

After the definitions of *v0* and *v1*, define the algebraic variable *vmax* for the largest value, and a selector *vs*

```
self.vmax = Algeb(v_str='maximum(v0, v1)',
                  tex_name='v_{max}',
                  e_str='vs_s0 * v0 + vs_s1 * v1 - vmax')

self.vs = Selector(self.v0, self.v1, fun=np.maximum.reduce)
```

The initial value of *vmax* is calculated by `maximum(v0, v1)`, which is the element-wise maximum in SymPy and will be generated into `np.maximum(v0, v1)`. The equation of *vmax* is to select the values based on *vs_s0* and *vs_s1*.

class andes.core.discrete.**Switcher**(*u*, *options: list | tuple*, *info: str = None*, *name: str = None*, *tex_name: str = None*, *cache=True*)

Switcher based on an input parameter.

The switch class takes one v-provider, compares the input with each value in the option list, and exports one flag array for each option. The flags are 0-indexed.

Exported flags are named with *_s0*, *_s1*, ..., with a total number of *len(options)*. See the examples section.

Notes

Switches needs to be distinguished from Selector.

Switcher is for generating flags indicating option selection based on an input parameter. Selector is for generating flags at run time based on variable values and a selection function.

Examples

The IEEEST model takes an input for selecting the signal. Options are 1 through 6. One can construct

```
self.IC = NumParam(info='input code 1-6')  # input code
self.SW = Switcher(u=self.IC, options=[0, 1, 2, 3, 4, 5, 6])
```

If the IC values from the data file ends up being

```
self.IC.v = np.array([1, 2, 2, 4, 6])
```

Then, the exported flag arrays will be

```
{'IC_s0': np.array([0, 0, 0, 0, 0]),
 'IC_s1': np.array([1, 0, 0, 0, 0]),
 'IC_s2': np.array([0, 1, 1, 0, 0]),
 'IC_s3': np.array([0, 0, 0, 0, 0]),
 'IC_s4': np.array([0, 0, 0, 1, 0]),
 'IC_s5': np.array([0, 0, 0, 0, 0]),
 'IC_s6': np.array([0, 0, 0, 0, 1])
}
```

where *IC_s0* is used for padding so that following flags align with the options.

Example of `Switcher` for multi-mode selection:

```
from andes.core.discrete import Switcher

self.sw = Switcher(
    u=self.mode,     # Selection input
    options=[0, 1, 2]
)
# sw.s0 = 1 when mode == 0
# sw.s1 = 1 when mode == 1
# sw.s2 = 1 when mode == 2

# Use in equation
self.y = Algeb(e_str='K1*x*sw_s0 + K2*x*sw_s1 + K3*x*sw_s2 - y')
```

Deadband

class andes.core.discrete.**DeadBand**(*u*, *center*, *lower*, *upper*, *enable=True*, *equal=False*, *zu=0.0*, *zl=0.0*, *zi=1.0*, *name=None*, *tex_name=None*, *info=None*)

The basic deadband type.

Parameters

u
[NumParam] The pre-deadband input variable

center
[NumParam] Neutral value of the output

lower
[NumParam] Lower bound

upper
[NumParam] Upper bound

enable
[bool] Enabled if True; Disabled and works as a pass-through if False.

Notes

Input changes within a deadband will incur no output changes. This component computes and exports three flags.

Three flags computed from the current input:

- zl: True if the input is below the lower threshold
- zi: True if the input is within the deadband
- zu: True if is above the lower threshold

Initial condition:

All three flags are initialized to zero. All flags are updated during *check_var* when enabled. If the deadband component is not enabled, all of them will remain zero.

Examples

Exported deadband flags need to be used in the algebraic equation corresponding to the post-deadband variable. Assume the pre-deadband input variable is *var_in* and the post-deadband variable is *var_out*. First, define a deadband instance *db* in the model using

```
self.db = DeadBand(u=self.var_in, center=self.dbc,
                   lower=self.dbl, upper=self.dbu)
```

To implement a no-memory deadband whose output returns to center when the input is within the band, the equation for *var* can be written as

```
var_out.e_str = 'var_in * (1 - db_zi) + \
                 (dbc * db_zi) - var_out'
```

class andes.core.discrete.**DeadBandRT**(*u*, *center*, *lower*, *upper*, *enable=True*)

Deadband with flags for directions of return.

Parameters

u
[NumParam] The pre-deadband input variable

center
[NumParam] Neutral value of the output

lower
[NumParam] Lower bound

upper
[NumParam] Upper bound

enable
[bool] Enabled if True; Disabled and works as a pass-through if False.

Notes

Input changes within a deadband will incur no output changes. This component computes and exports five flags. The additional two flags on top of *DeadBand* indicate the direction of return:

- zur: True if the input is/has been within the deadband and was returned from the upper threshold
- zlr: True if the input is/has been within the deadband and was returned from the lower threshold

Initial condition:

All five flags are initialized to zero. All flags are updated during *check_var* when enabled. If the deadband component is not enabled, all of them will remain zero.

Examples

To implement a deadband whose output is pegged at the nearest deadband bounds, the equation for *var* can be provided as

```
var_out.e_str = 'var_in * (1 - db_zi) + \
                 dbl * db_zlr + \
                 dbu * db_zur - var_out'
```

Example:

```
from andes.core.discrete import DeadBand

self.db = DeadBand(
    u=self.error,
    center=0,
    lower=-0.01,
    upper=0.01
)
# db.zi = 1 when within dead band
# db.zl = 1 when below
# db.zu = 1 when above

# Zero output in dead band
self.y = Algeb(e_str='error * (1 - db_zi) - y')
```

Others

class andes.core.discrete.**Average**(*u*, *mode='step'*, *delay=0*, *name=None*, *tex_name=None*, *info=None*)

Compute the average value of a BaseVar over a period of time or a number of simulation steps.

Average is based on the memory implemented in the Delay class. The same modes as in Delay are supported.

The output of the Average class is named `<INSTANCE_NAME>_v`, where *<INSTANCE_NAME>* is the instance name of Average.

class andes.core.discrete.**Delay**(*u*, *mode='step'*, *delay=0*, *name=None*, *tex_name=None*, *info=None*)

The delay class.

Delay allows to impose a predefined and fixed "delay" (in either steps or seconds) for an input variable. The amount of delay must be a scalar and has to be given when instantiating the Delay class when defining the model.

Delay implements an internal memorize to store past variable values.

The default delay mode is *step* but can be set to *time*. In the *time* mode, the value at the `current time - delay` will be interpolated based on the two nearest times and values.

Delay can be applied to a state or an algebraic variable. The exported variable is named `<INSTANCE_NAME>_v`, where *<INSTANCE_NAME>* is the name of the Delay instance.

class andes.core.discrete.**Derivative**(*u*, *name=None*, *tex_name=None*, *info=None*)

Compute the derivative of a variable using numerical differentiation.

Derivative is based on the storage implemented in the Delay class. The delay is set to *1* step so that the current and the previous step are used.

A simple first order derivative is computed using `u(t) - u(t-1) / tstep`, where `tstep` is the current step size.

Derivative is intended to be used for algebraic variables because of discontinuity. It can be applied to a state variable, but one should instead implement the right-hand side equation of the state variable in an algebraic equation to obtain the accurate derivative.

Alternatively, the washout filter (`andes.core.block.Washout`) can be used to implement a numerically stable derivative.

The output of the Derivative class is named `<INSTANCE_NAME>_v` just like `Delay`.

class andes.core.discrete.**Sampling**(*u*, *interval=1.0*, *offset=0.0*, *name=None*, *tex_name=None*, *info=None*)

Sample and hold

Sample an input variable periodically at the given time interval and hold the value until the next sample time.

For example, this class can be used to implement a 4-second sampling of the AGC signal.

The output of *Sampling* is named `<INSTANCE_NAME>_v`, where *<INSTANCE_NAME>* is the Sampling instance name.

class andes.core.discrete.**ShuntAdjust**(*, *v*, *lower*, *upper*, *bsw*, *gsw*, *dt*, *u*, *enable=True*, *min_iter=2*, *err_tol=0.01*, *name=None*, *tex_name=None*, *info=None*, *no_warn=False*)

Class for adjusting switchable shunts.

Parameters

v

[BaseVar] Voltage measurement

lower
[BaseParam] Lower voltage bound

upper
[BaseParam] Upper voltage bound

bsw
[SwBlock] SwBlock instance for susceptance

gsw
[SwBlock] SwBlock instance for conductance

dt
[NumParam] Delay time

u
[NumParam] Connection status

min_iter
[int] Minimum iteration number to enable shunt switching

err_tol
[float] Minimum iteration tolerance to enable switching

Using Flags in Equations

Piecewise Linear

```
# Saturated output
self.y = Algeb(
    e_str='x * lim_zi + upper * lim_zu + lower * lim_zl - y'
)
```

Conditional Behavior

```
# Different gains based on mode
self.y = Algeb(
    e_str='K1 * x * sw_s0 + K2 * x * sw_s1 - y'
)
```

Dead Band Application

```
# Zero output in dead band
self.y = Algeb(
    e_str='error * (1 - db_zi) - y'
)
```

Naming Convention

Discrete component flags use the component name as prefix:

```
self.lim = Limiter(...)
# Access in equations as: lim_zi, lim_zl, lim_zu

self.VL = Limiter(...)
# Access as: VL_zi, VL_zl, VL_zu
```

Common Patterns

Voltage-Based Load Shedding

```
self.v_low = LessThan(u=self.v, bound=0.9)
self.P = Algeb(e_str='P0 * (1 - v_low_z) - P')
```

Governor with Anti-Windup

```
self.lim = AntiWindup(u=self.gate, lower=0, upper=1)
self.gate = State(e_str='...')  # Gate position
```

Multi-Mode Controller

```
self.mode = Switcher(u=self.ctrl_mode, options=[0, 1, 2])
self.output = Algeb(
    e_str='out1*mode_s0 + out2*mode_s1 + out3*mode_s2 - output'
)
```

See Also

- *Blocks* - Transfer function blocks (many use discrete internally)
- *Services* - Service components
- *DAE Formulation* - Handling discontinuities in DAE

4.3.5 Blocks

The block library contains commonly used blocks such as transfer functions and nonlinear functions. Variables and equations are pre-defined for blocks to be used as "lego pieces" for scripting DAE models.

Background

The base class for blocks is `andes.core.block.Block`.

All variables in a block must be defined as attributes in the constructor, just like variable definition in models. The difference is that the variables are "exported" from a block to the capturing model. All exported variables need to be placed in a dictionary, `self.vars` at the end of the block constructor.

Blocks can be nested as advanced usage. See the API documentation below for more details.

class `andes.core.block.`**`Block`**(*name: str | None = None*, *tex_name: str | None = None*, *info: str | None = None*, *namespace: str = 'local'*)

Base class for control blocks.

Blocks are meant to be instantiated as Model attributes to provide pre-defined equation sets. Subclasses must overload the *__init__* method to take custom inputs. Subclasses of Block must overload the *define* method to provide initialization and equation strings. Exported variables, services and blocks must be constructed into a dictionary `self.vars` at the end of the constructor.

Blocks can be nested. A block can have blocks but itself as attributes and therefore reuse equations. When a block has sub-blocks, the outer block must be constructed with a``name``.

Nested block works in the following way: the parent block modifies the sub-block's `name` attribute by prepending the parent block's name at the construction phase. The parent block then exports the sub-block as a whole. When the parent Model class picks up the block, it will recursively import the variables in the block and the sub-blocks correctly. See the example section for details.

Parameters

name
[str, optional] Block name

tex_name
[str, optional] Block LaTeX name

info
[str, optional] Block description.

namespace
[str, local or parent] Namespace of the exported elements. If 'local', the block name will be prepended by the parent. If 'parent', the original element name will be used when exporting.

> **Warning:** It is a good practice to avoid more than one level of nesting, to avoid multi-underscore variable names.

Examples

Example for two-level nested blocks. Suppose we have the following hierarchy

SomeModel instance M contains an instance of LeadLag block named A, which contains a Lag instance named B. Both A and B exports two variables `x` and `y`.

In the code for SomeModel, the following code is used to instantiate LeadLag

```
class SomeModel:
    def __init__(...)
        ... self.A = LeadLag(name='A',
                             u=self.foo1, T1=self.foo2, T2=self.foo3)
```

To use Lag in the LeadLag code, the following lines are found in the constructor of LeadLag

```
class LeadLag:
    def __init__(name, ...)
        ... self.B = Lag(u=self.y, K=self.K, T=self.T)

        # register `self.B` with the name `A` self.vars = {..., 'B':
        self.B}
```

When instantiating any block instance, its `__setattr__` function assigns names to exported variables and blocks. For the LeadLag instance with the name `A`, its member attribute `B` is assigned the name `A_B` by convention. That is, `A_B` will be set to *B.name*.

When A is picked up by `SomeModel.__setattr__`, B is captured from A's exports with the name `A_B`. Recursively, B's variables are exported, Recall that *B.name* is now `A_B`, following the naming rule (parent block's name + variable name), B's internal variables become `A_B_x` and `A_B_y`.

Again, the LeadLag instance name (`A.name` in this example) must be given when instantiating in *SomeModel*'s constructor to ensure correct name propagation. If there is more than one level of nesting, other than the terminal-level block, all names of the parent blocks must be provided at instantiation.

In such a way, B's `define()` needs no modification since the naming rule is the same. For example, B's internal y is always `{self.name}_y`, although the nested B has gotten a new name `A_B`.

define()
Function for setting the initialization and equation strings for internal variables. This method must be implemented by subclasses.

The equations should be written with the "final" variable names. Let's say the block instance is named *blk* (kept at `self.name` of the block), and an internal variable *v* is defined. The internal variable will be captured as `blk_v` by the parent model. Therefore, all equations should use `{self.name}_v` to represent variable `v`, where `{self.name}` is the name of the block at run time.

On the other hand, the names of externally provided parameters or variables are obtained by directly accessing the `name` attribute. For example, if `self.T` is a parameter provided through the block constructor, `{self.T.name}` should be used in the equation.

See also:

`PIController.define`
Equations for the PI Controller block

Examples

An internal variable `v` has a trivial equation `T = v`, where T is a parameter provided to the block constructor.

In the model, one has

```
class SomeModel():
    def __init__(...)
        self.input = Algeb()
        self.T = Param()

        self.blk = ExampleBlock(u=self.input, T=self.T)
```

In the ExampleBlock function, the internal variable is defined in the constructor as

```
class ExampleBlock():
    def __init__(...):
        self.v = Algeb()
        self.vars = {'v', self.v}
```

In the `define`, the equation is provided as

```
def define(self):
    self.v.v_str = '{self.T.name}'
    self.v.e_str = '{self.T.name} - {self.name}_v'
```

In the parent model, `v` from the block will be captured as `blk_v`, and the equation will evaluate into

```
self.blk_v.v_str = 'T'
self.blk_v.e_str = 'T - blk_v'
```

Block Types

`Block`([name, tex_name, info, namespace])	Base class for control blocks.
`Gain`(u, K[, name, tex_name, info])	Gain block.
`GainLimiter`(u, K, R, lower, upper[, ...])	Gain followed by a limiter and another gain.
`Piecewise`(u, points, funs[, name, tex_name, ...])	Piecewise block.
`HVGate`(u1, u2[, name, tex_name, info])	High Value Gate.
`LVGate`(u1, u2[, name, tex_name, info])	Low Value Gate.
`DeadBand1`(u, center, lower, upper[, gain, ...])	Deadband type 1 (linear, non-step).
`Integrator`(u, T, K, y0[, check_init, name, ...])	Integrator block.
`IntegratorAntiWindup`(u, T, K, y0, lower, upper)	Integrator block with anti-windup limiter.
`Lag`(u, T, K[, D, name, tex_name, info])	Lag (low pass filter) transfer function.
`LagAntiWindup`(u, T, K, lower, upper[, D, ...])	Lag (low pass filter) transfer function block with an anti-windup limiter.
`LagFreeze`(u, T, K, freeze[, D, name, ...])	Lag with an input to freeze the state.
`LagAWFreeze`(u, T, K, lower, upper, freeze[, ...])	Lag with anti-windup limiter and state freeze.
`LagRate`(u, T, K, rate_lower, rate_upper[, ...])	Lag (low pass filter) transfer function block with a rate limiter.
`LagAntiWindupRate`(u, T, K, lower, upper, ...)	Lag (low pass filter) transfer function block with a rate limiter and an anti-windup limiter.
`Washout`(u, T, K[, name, tex_name, info])	Washout filter (high pass) block.
`WashoutOrLag`(u, T, K[, name, zero_out, ...])	Washout with the capability to convert to Lag when K = 0.
`LeadLag`(u, T1, T2[, K, zero_out, name, ...])	Lead-Lag transfer function block in series implementation.
`LeadLagLimit`(u, T1, T2, lower, upper[, ...])	Lead-Lag transfer function block with hard limiter (series implementation).
`Lag2ndOrd`(u, K, T1, T2[, name, tex_name, info])	Second order lag transfer function (low-pass filter).
`LeadLag2ndOrd`(u, T1, T2, T3, T4[, zero_out, ...])	Second-order lead-lag transfer function block.
`PIController`(u, kp, ki[, ref, x0, name, ...])	Proportional Integral Controller.
`PIAWHardLimit`(u, kp, ki, aw_lower, aw_upper, ...)	PI controller with anti-windup limiter on the integrator and hard limit on the output.
`PITrackAW`(u, kp, ki, ks, lower, upper[, ...])	PI with tracking anti-windup limiter
`PIFreeze`(u, kp, ki, freeze[, ref, x0, name, ...])	PI controller with state freeze.
`PITrackAWFreeze`(u, kp, ki, ks, lower, upper, ...)	PI controller with tracking anti-windup limiter and state freeze.
`PIDController`(u, kp, ki, kd, Td, name[, ...])	Proportional Integral Derivative Controller.
`PIDAWHardLimit`(u, kp, ki, kd, Td, aw_lower, ...)	PID controller with anti-windup limiter on the integrator and hard limit on the output.
`PIDTrackAW`(u, kp, ki, kd, Td, ks, lower, upper)	PID with tracking anti-windup limiter

andes.core.block.Block

class andes.core.block.**Block**(*name: str | None = None, tex_name: str | None = None, info: str | None = None, namespace: str = 'local'*)

Base class for control blocks.

Blocks are meant to be instantiated as Model attributes to provide pre-defined equation sets. Subclasses must overload the *__init__* method to take custom inputs. Subclasses of Block must overload the *define* method to provide initialization and equation strings. Exported variables, services and blocks must be constructed into a dictionary `self.vars` at the end of the constructor.

Blocks can be nested. A block can have blocks but itself as attributes and therefore reuse equations. When a block has sub-blocks, the outer block must be constructed with a``name``.

Nested block works in the following way: the parent block modifies the sub-block's `name` attribute by prepending the parent block's name at the construction phase. The parent block then exports the sub-block as a whole. When the parent Model class picks up the block, it will recursively import the variables in the block and the sub-blocks correctly. See the example section for details.

Parameters

name
[str, optional] Block name

tex_name
[str, optional] Block LaTeX name

info
[str, optional] Block description.

namespace
[str, local or parent] Namespace of the exported elements. If 'local', the block name will be prepended by the parent. If 'parent', the original element name will be used when exporting.

> **Warning:** It is a good practice to avoid more than one level of nesting, to avoid multi-underscore variable names.

Examples

Example for two-level nested blocks. Suppose we have the following hierarchy

SomeModel instance M contains an instance of LeadLag block named A, which contains a Lag instance named B. Both A and B exports two variables `x` and `y`.

In the code for SomeModel, the following code is used to instantiate LeadLag

```
class SomeModel:
    def __init__(...)
```

(continues on next page)

(continued from previous page)

```
        ... self.A = LeadLag(name='A',
                             u=self.foo1, T1=self.foo2, T2=self.foo3)
```

To use Lag in the LeadLag code, the following lines are found in the constructor of LeadLag

```
class LeadLag:
    def __init__(name, ...)
        ... self.B = Lag(u=self.y, K=self.K, T=self.T)

        # register `self.B` with the name `A` self.vars = {..., 'B':
        self.B}
```

When instantiating any block instance, its `__setattr__` function assigns names to exported variables and blocks. For the LeadLag instance with the name `A`, its member attribute `B` is assigned the name `A_B` by convention. That is, `A_B` will be set to *B.name*.

When A is picked up by `SomeModel.__setattr__`, B is captured from A's exports with the name `A_B`. Recursively, B's variables are exported, Recall that *B.name* is now `A_B`, following the naming rule (parent block's name + variable name), B's internal variables become `A_B_x` and `A_B_y`.

Again, the LeadLag instance name (`A.name` in this example) must be given when instantiating in *SomeModel*'s constructor to ensure correct name propagation. If there is more than one level of nesting, other than the terminal-level block, all names of the parent blocks must be provided at instantiation.

In such a way, B's `define()` needs no modification since the naming rule is the same. For example, B's internal y is always `{self.name}_y`, although the nested B has gotten a new name `A_B`.

__init__(*name: str | None = None*, *tex_name: str | None = None*, *info: str | None = None*, *namespace: str = 'local'*)

Methods

`define()`	Function for setting the initialization and equation strings for internal variables.
`enforce_tex_name(fields)`	Enforce tex_name is not None
`export()`	Method for exporting instances defined in this class in a dictionary.
`f_numeric(**kwargs)`	Function call to update differential equation values.
`g_numeric(**kwargs)`	Function call to update algebraic equation values.
`j_numeric(**kwargs)`	Per-iteration numerical Jacobian update.
`j_reset()`	Helper function to clear the lists holding the numerical Jacobians.
`j_setup()`	One-time Jacobian sparsity pattern and constant value setup.

Attributes

class_name	Return the class name.

andes.core.block.Gain

class andes.core.block.**Gain**(*u*, *K*, *name=None*, *tex_name=None*, *info=None*)

Gain block.

```
      ┌───┐
u -> │ K │ -> y
      └───┘
```

Exports an algebraic output *y*.

__init__(*u*, *K*, *name=None*, *tex_name=None*, *info=None*)

Methods

define()	Implemented equation and the initial condition are
enforce_tex_name(fields)	Enforce tex_name is not None
export()	Method for exporting instances defined in this class in a dictionary.
f_numeric(**kwargs)	Function call to update differential equation values.
g_numeric(**kwargs)	Function call to update algebraic equation values.
j_numeric(**kwargs)	Per-iteration numerical Jacobian update.
j_reset()	Helper function to clear the lists holding the numerical Jacobians.
j_setup()	One-time Jacobian sparsity pattern and constant value setup.

Attributes

class_name	Return the class name.

andes.core.block.GainLimiter

class andes.core.block.**GainLimiter**(*u*, *K*, *R*, *lower*, *upper*, *no_lower=False*, *no_upper=False*, *sign_lower=1*, *sign_upper=1*, *allow_adjust=True*, *name=None*, *tex_name=None*, *info=None*)

Gain followed by a limiter and another gain.

Exports the limited output *y*, unlimited output *x*, and HardLimiter *lim*.

```
     ┌─────┐          upper   ┌─────┐
     │     │         /-----   │     │
u -> │  K  | -> x  /   ->     |  R  | -> y
     │     | _____/           |     |
     └─────┘  lower           └─────┘
```

Parameters

u
[str, BaseVar] Input variable, or an equation string for constructing an anonymous variable

K
[str, BaseParam, BaseService] Initial gain for *u* before limiter

R
[str, BaseParam, BaseService] Post limiter gain

__init__(*u*, *K*, *R*, *lower*, *upper*, *no_lower=False*, *no_upper=False*, *sign_lower=1*, *sign_upper=1*, *allow_adjust=True*, *name=None*, *tex_name=None*, *info=None*)

Methods

define()	TODO: write docstring
enforce_tex_name(fields)	Enforce tex_name is not None
export()	Method for exporting instances defined in this class in a dictionary.
f_numeric(**kwargs)	Function call to update differential equation values.
g_numeric(**kwargs)	Function call to update algebraic equation values.
j_numeric(**kwargs)	Per-iteration numerical Jacobian update.
j_reset()	Helper function to clear the lists holding the numerical Jacobians.
j_setup()	One-time Jacobian sparsity pattern and constant value setup.

Attributes

`class_name`	Return the class name.

andes.core.block.Piecewise

class `andes.core.block.`**`Piecewise`**(*u*, *points: List | Tuple*, *funs: List | Tuple*, *name=None*, *tex_name=None*, *info=None*)

Piecewise block. Outputs an algebraic variable *y*.

This block takes a list of *N* points, *[x0, x1, ...x_{n-1}]* to define N+1 ranges, namely (-inf, x0), (x0, x1), ..., (x_{n-1}, +inf). and a list of *N+1* function strings *[fun0, ..., fun_n]*.

Inputs that fall within each range applies the corresponding function. The first range (-inf, x0) applies *fun_0*, and the last range (x_{n-1}, +inf) applies the last function *fun_n*.

The function returns zero if no condition is met.

Note: Piecewise.y must remain `Algeb` (not `Observable`) because: (1) model authors may set `v_iter` on the output for coupled iterative initialization (e.g., EXAC1), and (2) zero-valued branches cause `ComplexInfinity` when the expression is substituted into denominators during code generation.

Parameters

points
[list, tuple] A list of piecewise points. Need to be provided in the constructor function.

funs
[list, tuple] A list of strings for the piecewise functions. Need to be provided in the overloaded *define* function.

`__init__`(*u*, *points: List | Tuple*, *funs: List | Tuple*, *name=None*, *tex_name=None*, *info=None*)

Methods

`define()`	Build the equation string for the piecewise equations.
`enforce_tex_name(fields)`	Enforce tex_name is not None
`export()`	Method for exporting instances defined in this class in a dictionary.
`f_numeric(**kwargs)`	Function call to update differential equation values.
`g_numeric(**kwargs)`	Function call to update algebraic equation values.
`j_numeric(**kwargs)`	Per-iteration numerical Jacobian update.
`j_reset()`	Helper function to clear the lists holding the numerical Jacobians.
`j_setup()`	One-time Jacobian sparsity pattern and constant value setup.

Attributes

`class_name`	Return the class name.

andes.core.block.HVGate

class `andes.core.block.`**`HVGate`**(*u1*, *u2*, *name=None*, *tex_name=None*, *info=None*)

High Value Gate. Outputs the maximum of two inputs.

```
      ┌──────────┐
u1 -> │ HV Gate  │
      │          │ ->  y
u2 -> │   (MAX)  │
      └──────────┘
```

`__init__`(*u1*, *u2*, *name=None*, *tex_name=None*, *info=None*)

Methods

`define()`	Implemented equations and initial conditions
`enforce_tex_name`(fields)	Enforce tex_name is not None
`export()`	Method for exporting instances defined in this class in a dictionary.
`f_numeric`(**kwargs)	Function call to update differential equation values.
`g_numeric`(**kwargs)	Function call to update algebraic equation values.
`j_numeric`(**kwargs)	Per-iteration numerical Jacobian update.
`j_reset()`	Helper function to clear the lists holding the numerical Jacobians.
`j_setup()`	One-time Jacobian sparsity pattern and constant value setup.

Attributes

`class_name`	Return the class name.

andes.core.block.LVGate

class `andes.core.block.`**`LVGate`**(*u1*, *u2*, *name=None*, *tex_name=None*, *info=None*)

Low Value Gate. Outputs the minimum of the two inputs.

```
      ┌──────────┐
u1 -> │ LV Gate  │
      │          │ ->  y
u2 -> │   (MIN)  │
      └──────────┘
```

`__init__`(*u1*, *u2*, *name=None*, *tex_name=None*, *info=None*)

Methods

`define()`	Implemented equations and initial conditions
`enforce_tex_name`(fields)	Enforce tex_name is not None
`export()`	Method for exporting instances defined in this class in a dictionary.
`f_numeric`(**kwargs)	Function call to update differential equation values.
`g_numeric`(**kwargs)	Function call to update algebraic equation values.
`j_numeric`(**kwargs)	Per-iteration numerical Jacobian update.
`j_reset()`	Helper function to clear the lists holding the numerical Jacobians.
`j_setup()`	One-time Jacobian sparsity pattern and constant value setup.

Attributes

`class_name`	Return the class name.

andes.core.block.DeadBand1

class `andes.core.block.`**`DeadBand1`**(*u*, *center*, *lower*, *upper*, *gain=1.0*, *enable=True*, *name=None*, *tex_name=None*, *info=None*, *namespace='local'*)

Deadband type 1 (linear, non-step).

Note: DeadBand1.y must remain `Algeb` (not `Observable`) because other models (e.g., DGPRCT1) reference it via `ExtAlgeb`.

Parameters

center
Default value when within the deadband. If the input is an error signal, center should be set to zero.

gain
Gain multiplied to DeadBand discrete block's output.

Notes

Block diagram

```
      |   /
______|__/___   -> Gain -> DeadBand1_y
   /  |
  /   |
```

__init__(*u*, *center*, *lower*, *upper*, *gain=1.0*, *enable=True*, *name=None*, *tex_name=None*, *info=None*, *namespace='local'*)

Methods

define()	Notes
enforce_tex_name(fields)	Enforce tex_name is not None
export()	Method for exporting instances defined in this class in a dictionary.
f_numeric(**kwargs)	Function call to update differential equation values.
g_numeric(**kwargs)	Function call to update algebraic equation values.
j_numeric(**kwargs)	Per-iteration numerical Jacobian update.
j_reset()	Helper function to clear the lists holding the numerical Jacobians.
j_setup()	One-time Jacobian sparsity pattern and constant value setup.

Attributes

class_name	Return the class name.

andes.core.block.Integrator

class andes.core.block.**Integrator**(*u*, *T*, *K*, *y0*, *check_init=True*, *name=None*, *tex_name=None*, *info=None*)

Integrator block.

```
      ┌─────┐
u ->  │ K/sT │ -> y
      └─────┘
```

Exports a differential variable *y*.

The initial output needs to be specified through *y0*.

__init__(*u*, *T*, *K*, *y0*, *check_init=True*, *name=None*, *tex_name=None*, *info=None*)

Methods

`define()`	Implemented equation and the initial condition are
`enforce_tex_name(fields)`	Enforce tex_name is not None
`export()`	Method for exporting instances defined in this class in a dictionary.
`f_numeric(**kwargs)`	Function call to update differential equation values.
`g_numeric(**kwargs)`	Function call to update algebraic equation values.
`j_numeric(**kwargs)`	Per-iteration numerical Jacobian update.
`j_reset()`	Helper function to clear the lists holding the numerical Jacobians.
`j_setup()`	One-time Jacobian sparsity pattern and constant value setup.

Attributes

`class_name`	Return the class name.

andes.core.block.IntegratorAntiWindup

class andes.core.block.**IntegratorAntiWindup**(*u*, *T*, *K*, *y0*, *lower*, *upper*, *name=None*, *tex_name=None*, *info=None*, *no_warn=False*)

Integrator block with anti-windup limiter.

```
           upper
          /-----
      ┌─────┐
u ->  │ K/sT │ -> y
      └─────┘
```

(continues on next page)

(continued from previous page)

```
  _____/
  lower
```

Exports a differential variable *y* and an AntiWindup *lim*. The initial output must be specified through *y0*.

__init__(*u*, *T*, *K*, *y0*, *lower*, *upper*, *name=None*, *tex_name=None*, *info=None*, *no_warn=False*)

Methods

`define()`	Implemented equation and the initial condition are
`enforce_tex_name`(fields)	Enforce tex_name is not None
`export()`	Method for exporting instances defined in this class in a dictionary.
`f_numeric`(**kwargs)	Function call to update differential equation values.
`g_numeric`(**kwargs)	Function call to update algebraic equation values.
`j_numeric`(**kwargs)	Per-iteration numerical Jacobian update.
`j_reset()`	Helper function to clear the lists holding the numerical Jacobians.
`j_setup()`	One-time Jacobian sparsity pattern and constant value setup.

Attributes

`class_name`	Return the class name.

andes.core.block.Lag

class andes.core.block.**Lag**(*u*, *T*, *K*, *D=1*, *name=None*, *tex_name=None*, *info=None*)

Lag (low pass filter) transfer function.

```
      ┌─────────┐
      │    K    │
u ->  │ ─────── │ -> y
      │ D + sT  │
      └─────────┘
```

Exports one state variable *y* as the output.

Parameters

K
Gain

T
Time constant

D
Constant

u
Input variable

__init__(*u*, *T*, *K*, *D=1*, *name=None*, *tex_name=None*, *info=None*)

Methods

`define()`	
	Notes
`enforce_tex_name`(fields)	Enforce tex_name is not None
`export()`	Method for exporting instances defined in this class in a dictionary.
`f_numeric`(**kwargs)	Function call to update differential equation values.
`g_numeric`(**kwargs)	Function call to update algebraic equation values.
`j_numeric`(**kwargs)	Per-iteration numerical Jacobian update.
`j_reset()`	Helper function to clear the lists holding the numerical Jacobians.
`j_setup()`	One-time Jacobian sparsity pattern and constant value setup.

Attributes

`class_name`	Return the class name.

andes.core.block.LagAntiWindup

class andes.core.block.**LagAntiWindup**(*u*, *T*, *K*, *lower*, *upper*, *D=1*, *name=None*, *tex_name=None*, *info=None*)

Lag (low pass filter) transfer function block with an anti-windup limiter.

```
             upper
           /------
      ┌─────────┐
      │    K    │
u ->  │ ─────── │ -> y
      │ D + sT  │
      └─────────┘
   ______/
   lower
```

Exports one state variable *y* as the output and one AntiWindup instance *lim*.

Parameters

K
Gain

T
Time constant

D
Constant

u
Input variable

__init__(*u*, *T*, *K*, *lower*, *upper*, *D=1*, *name=None*, *tex_name=None*, *info=None*)

Methods

`define()`	
	Notes
`enforce_tex_name`(fields)	Enforce tex_name is not None
`export()`	Method for exporting instances defined in this class in a dictionary.
`f_numeric`(**kwargs)	Function call to update differential equation values.
`g_numeric`(**kwargs)	Function call to update algebraic equation values.
`j_numeric`(**kwargs)	Per-iteration numerical Jacobian update.
`j_reset()`	Helper function to clear the lists holding the numerical Jacobians.
`j_setup()`	One-time Jacobian sparsity pattern and constant value setup.

Attributes

`class_name`	Return the class name.

andes.core.block.LagFreeze

class andes.core.block.**LagFreeze**(*u*, *T*, *K*, *freeze*, *D=1*, *name=None*, *tex_name=None*, *info=None*)

Lag with an input to freeze the state.

During the period when the freeze signal is 1, the LagFreeze output will be frozen.

__init__(*u*, *T*, *K*, *freeze*, *D=1*, *name=None*, *tex_name=None*, *info=None*)

Methods

`define()`	
	Notes
`enforce_tex_name`(fields)	Enforce tex_name is not None
`export()`	Method for exporting instances defined in this class in a dictionary.
`f_numeric`(**kwargs)	Function call to update differential equation values.
`g_numeric`(**kwargs)	Function call to update algebraic equation values.
`j_numeric`(**kwargs)	Per-iteration numerical Jacobian update.
`j_reset()`	Helper function to clear the lists holding the numerical Jacobians.
`j_setup()`	One-time Jacobian sparsity pattern and constant value setup.

Attributes

`class_name`	Return the class name.

andes.core.block.LagAWFreeze

class andes.core.block.**LagAWFreeze**(*u*, *T*, *K*, *lower*, *upper*, *freeze*, *D=1*, *name=None*, *tex_name=None*, *info=None*)

Lag with anti-windup limiter and state freeze.

Note that the output *y* is a state variable.

__init__(*u*, *T*, *K*, *lower*, *upper*, *freeze*, *D=1*, *name=None*, *tex_name=None*, *info=None*)

Methods

`define()`	
	Notes
`enforce_tex_name`(fields)	Enforce tex_name is not None
`export()`	Method for exporting instances defined in this class in a dictionary.
`f_numeric`(**kwargs)	Function call to update differential equation values.
`g_numeric`(**kwargs)	Function call to update algebraic equation values.
`j_numeric`(**kwargs)	Per-iteration numerical Jacobian update.
`j_reset()`	Helper function to clear the lists holding the numerical Jacobians.
`j_setup()`	One-time Jacobian sparsity pattern and constant value setup.

Attributes

`class_name`	Return the class name.

andes.core.block.LagRate

class andes.core.block.**LagRate**(*u*, *T*, *K*, *rate_lower*, *rate_upper*, *D=1*, *rate_no_lower=False*, *rate_no_upper=False*, *rate_lower_cond=None*, *rate_upper_cond=None*, *name=None*, *tex_name=None*, *info=None*)

Lag (low pass filter) transfer function block with a rate limiter.

```
             / rate_upper
        ┌─────────┐
        │    K    │
  u ->  │ ─────── │ -> y
        │ D + sT  │
        └─────────┘
rate_lower /
```

Exports one state variable *y* as the output and one AntiWindupRate instance *lim*.

Parameters

K

Gain

T
Time constant

D
Constant

u
Input variable

__init__(*u*, *T*, *K*, *rate_lower*, *rate_upper*, *D=1*, *rate_no_lower=False*, *rate_no_upper=False*, *rate_lower_cond=None*, *rate_upper_cond=None*, *name=None*, *tex_name=None*, *info=None*)

Methods

	Notes
`define()`	
`enforce_tex_name`(fields)	Enforce tex_name is not None
`export()`	Method for exporting instances defined in this class in a dictionary.
`f_numeric`(**kwargs)	Function call to update differential equation values.
`g_numeric`(**kwargs)	Function call to update algebraic equation values.
`j_numeric`(**kwargs)	Per-iteration numerical Jacobian update.
`j_reset()`	Helper function to clear the lists holding the numerical Jacobians.
`j_setup()`	One-time Jacobian sparsity pattern and constant value setup.

Attributes

`class_name`	Return the class name.

andes.core.block.LagAntiWindupRate

class andes.core.block.**LagAntiWindupRate**(*u*, *T*, *K*, *lower*, *upper*, *rate_lower*, *rate_upper*, *D=1*, *no_lower=False*, *no_upper=False*, *rate_no_lower=False*, *rate_no_upper=False*, *rate_lower_cond=None*, *rate_upper_cond=None*, *name=None*, *tex_name=None*, *info=None*)

Lag (low pass filter) transfer function block with a rate limiter and an anti-windup limiter.

```
             upper
rate_upper /------
     ┌──────────┐
     │    K     │
u -> │ ──────── │ -> y
     │ D + sT   │
     └──────────┘
   ______/ rate_lower
   lower
```

Exports one state variable *y* as the output and one AntiWindupRate instance *lim*.

Parameters

K
Gain

T
Time constant

D
Constant

u
Input variable

__init__(*u*, *T*, *K*, *lower*, *upper*, *rate_lower*, *rate_upper*, *D=1*, *no_lower=False*, *no_upper=False*, *rate_no_lower=False*, *rate_no_upper=False*, *rate_lower_cond=None*, *rate_upper_cond=None*, *name=None*, *tex_name=None*, *info=None*)

Methods

	Notes
`define()`	
`enforce_tex_name`(fields)	Enforce tex_name is not None
`export()`	Method for exporting instances defined in this class in a dictionary.
`f_numeric`(**kwargs)	Function call to update differential equation values.
`g_numeric`(**kwargs)	Function call to update algebraic equation values.
`j_numeric`(**kwargs)	Per-iteration numerical Jacobian update.
`j_reset()`	Helper function to clear the lists holding the numerical Jacobians.
`j_setup()`	One-time Jacobian sparsity pattern and constant value setup.

Attributes

`class_name`	Return the class name.

andes.core.block.Washout

class `andes.core.block.`**`Washout`**(*u*, *T*, *K*, *name=None*, *tex_name=None*, *info=None*)

Washout filter (high pass) block.

```
     ┌─────────┐
     │   sK    │
u -> │ ─────── │ -> y
     │ 1 + sT  │
     └─────────┘
```

Exports state x (symbol x') and output algebraic variable y.

`__init__`(*u*, *T*, *K*, *name=None*, *tex_name=None*, *info=None*)

Methods

`define()`	
	Notes
`enforce_tex_name`(fields)	Enforce tex_name is not None
`export()`	Method for exporting instances defined in this class in a dictionary.
`f_numeric`(**kwargs)	Function call to update differential equation values.
`g_numeric`(**kwargs)	Function call to update algebraic equation values.
`j_numeric`(**kwargs)	Per-iteration numerical Jacobian update.
`j_reset()`	Helper function to clear the lists holding the numerical Jacobians.
`j_setup()`	One-time Jacobian sparsity pattern and constant value setup.

Attributes

class_name	Return the class name.

andes.core.block.WashoutOrLag

class andes.core.block.**WashoutOrLag**(*u*, *T*, *K*, *name=None*, *zero_out=True*, *tex_name=None*, *info=None*)

Washout with the capability to convert to Lag when K = 0.

Can be enabled with *zero_out*. Need to provide *name* to construct.

Exports state x (symbol x'), output algebraic variable y, and a LessThan block LT.

Parameters

zero_out
[bool, optional] If True, `sT` will become 1, and the washout will become a low-pass filter. If False, functions as a regular Washout.

__init__(*u*, *T*, *K*, *name=None*, *zero_out=True*, *tex_name=None*, *info=None*)

Methods

define()	
	Notes
enforce_tex_name(fields)	Enforce tex_name is not None
export()	Method for exporting instances defined in this class in a dictionary.
f_numeric(**kwargs)	Function call to update differential equation values.
g_numeric(**kwargs)	Function call to update algebraic equation values.
j_numeric(**kwargs)	Per-iteration numerical Jacobian update.
j_reset()	Helper function to clear the lists holding the numerical Jacobians.
j_setup()	One-time Jacobian sparsity pattern and constant value setup.

Attributes

`class_name`	Return the class name.

andes.core.block.LeadLag

class `andes.core.block.`**`LeadLag`**(*u*, *T1*, *T2*, *K=1*, *zero_out=True*, *name=None*, *tex_name=None*, *info=None*)

Lead-Lag transfer function block in series implementation.

```
      ┌─────────────┐
      │   1 + sT1   │
u ->  │ K ───────── │ -> y
      │   1 + sT2   │
      └─────────────┘
```

Exports two variables: internal state `x` and output algebraic variable `y`.

Parameters

u
[BaseVar or BaseParam] Input signal.

T1
[BaseParam] Numerator (lead) time constant.

T2
[BaseParam] Denominator (lag) time constant. Also used as the `t_const` of the internal state `x`.

K
[BaseParam or numeric, optional] Static gain (default 1).

zero_out
[bool, optional] If `True` (default), bypass the block as `y = K * u` when T2 <= 0. Set to `False` only if T2 > 0 is guaranteed by the data.

`__init__`(*u*, *T1*, *T2*, *K=1*, *zero_out=True*, *name=None*, *tex_name=None*, *info=None*)

Methods

`define()`	
	Notes
`enforce_tex_name`(fields)	Enforce tex_name is not None
`export()`	Method for exporting instances defined in this class in a dictionary.
`f_numeric`(**kwargs)	Function call to update differential equation values.
`g_numeric`(**kwargs)	Function call to update algebraic equation values.
`j_numeric`(**kwargs)	Per-iteration numerical Jacobian update.
`j_reset()`	Helper function to clear the lists holding the numerical Jacobians.
`j_setup()`	One-time Jacobian sparsity pattern and constant value setup.

Attributes

`class_name`	Return the class name.

andes.core.block.LeadLagLimit

class `andes.core.block.`**`LeadLagLimit`**(*u*, *T1*, *T2*, *lower*, *upper*, *name=None*, *tex_name=None*, *info=None*)

Lead-Lag transfer function block with hard limiter (series implementation).

```
     ┌──────────┐           upper
     │ 1 + sT1  │          /-----
u -> │ ──────── │ -> ynl  / -> y
     │ 1 + sT2  │  _____/
     └──────────┘  lower
```

Exports four variables: state *x*, output before hard limiter *ynl*, output *y*, and AntiWindup *lim*.

`__init__`(*u*, *T1*, *T2*, *lower*, *upper*, *name=None*, *tex_name=None*, *info=None*)

Methods

define()	Notes
enforce_tex_name(fields)	Enforce tex_name is not None
export()	Method for exporting instances defined in this class in a dictionary.
f_numeric(**kwargs)	Function call to update differential equation values.
g_numeric(**kwargs)	Function call to update algebraic equation values.
j_numeric(**kwargs)	Per-iteration numerical Jacobian update.
j_reset()	Helper function to clear the lists holding the numerical Jacobians.
j_setup()	One-time Jacobian sparsity pattern and constant value setup.

Attributes

class_name	Return the class name.

andes.core.block.Lag2ndOrd

class andes.core.block.**Lag2ndOrd**(*u*, *K*, *T1*, *T2*, *name=None*, *tex_name=None*, *info=None*)

Second order lag transfer function (low-pass filter).

```
      ┌──────────────────┐
      │         K        │
u ->  │ ─────────────────── │ -> y
      │ 1 + sT1 + s^2 T2 │
      └──────────────────┘
```

Exports one two state variables (x, y), where y is the output.

Parameters

u
Input

K
Gain

T1
First order time constant

T2
Second order time constant

__init__(*u*, *K*, *T1*, *T2*, *name=None*, *tex_name=None*, *info=None*)

Methods

	Notes
define()	
enforce_tex_name(fields)	Enforce tex_name is not None
export()	Method for exporting instances defined in this class in a dictionary.
f_numeric(**kwargs)	Function call to update differential equation values.
g_numeric(**kwargs)	Function call to update algebraic equation values.
j_numeric(**kwargs)	Per-iteration numerical Jacobian update.
j_reset()	Helper function to clear the lists holding the numerical Jacobians.
j_setup()	One-time Jacobian sparsity pattern and constant value setup.

Attributes

class_name	Return the class name.

andes.core.block.LeadLag2ndOrd

class andes.core.block.**LeadLag2ndOrd**(*u*, *T1*, *T2*, *T3*, *T4*, *zero_out=False*, *name=None*, *tex_name=None*, *info=None*)

Second-order lead-lag transfer function block.

```
     ┌─────────────────┐
     │ 1 + sT3 + s^2 T4 │
u -> │ ─────────────────── │ -> y
     │ 1 + sT1 + s^2 T2 │
     └─────────────────┘
```

Exports two internal states (`x1` and `x2`) and output algebraic variable `y`.

Parameters

u
[BaseVar or BaseParam] Input signal.

T1
[BaseParam] Denominator first-order time constant.

T2
[BaseParam] Denominator second-order time constant. Also used as the `t_const` of the internal state `x1`.

T3
[BaseParam] Numerator first-order time constant.

T4
[BaseParam] Numerator second-order time constant.

zero_out
[bool, optional] If `True`, bypass the block as `y = u` when T2 <= 0. Default is `False`.

`__init__`(*u*, *T1*, *T2*, *T3*, *T4*, *zero_out=False*, *name=None*, *tex_name=None*, *info=None*)

Methods

	Notes
`define()`	
`enforce_tex_name`(fields)	Enforce tex_name is not None
`export()`	Method for exporting instances defined in this class in a dictionary.
`f_numeric`(**kwargs)	Function call to update differential equation values.
`g_numeric`(**kwargs)	Function call to update algebraic equation values.
`j_numeric`(**kwargs)	Per-iteration numerical Jacobian update.
`j_reset()`	Helper function to clear the lists holding the numerical Jacobians.
`j_setup()`	One-time Jacobian sparsity pattern and constant value setup.

Attributes

class_name	Return the class name.

andes.core.block.PIController

class andes.core.block.**PIController**(*u*, *kp*, *ki*, *ref=0.0*, *x0=0.0*, *name=None*, *tex_name=None*, *info=None*)

Proportional Integral Controller.

The controller takes an error signal as the input. It takes an optional *ref* signal, which will be subtracted from the input.

Parameters

u
[BaseVar] The input variable instance

kp
[BaseParam] The proportional gain parameter instance

ki
[[type]] The integral gain parameter instance

__init__(*u*, *kp*, *ki*, *ref=0.0*, *x0=0.0*, *name=None*, *tex_name=None*, *info=None*)

Methods

define()	Define equations for the PI Controller.
enforce_tex_name(fields)	Enforce tex_name is not None
export()	Method for exporting instances defined in this class in a dictionary.
f_numeric(**kwargs)	Function call to update differential equation values.
g_numeric(**kwargs)	Function call to update algebraic equation values.
j_numeric(**kwargs)	Per-iteration numerical Jacobian update.
j_reset()	Helper function to clear the lists holding the numerical Jacobians.
j_setup()	One-time Jacobian sparsity pattern and constant value setup.

Attributes

class_name	Return the class name.

andes.core.block.PIAWHardLimit

class andes.core.block.**PIAWHardLimit**(*u*, *kp*, *ki*, *aw_lower*, *aw_upper*, *lower*, *upper*, *no_lower=False*, *no_upper=False*, *ref=0.0*, *x0=0.0*, *name=None*, *tex_name=None*, *info=None*)

PI controller with anti-windup limiter on the integrator and hard limit on the output.

Limits `lower` and `upper` are on the final output, and `aw_lower` `aw_upper` are on the integrator.

__init__(*u*, *kp*, *ki*, *aw_lower*, *aw_upper*, *lower*, *upper*, *no_lower=False*, *no_upper=False*, *ref=0.0*, *x0=0.0*, *name=None*, *tex_name=None*, *info=None*)

Methods

define()	Define equations for the PI Controller.
enforce_tex_name(fields)	Enforce tex_name is not None
export()	Method for exporting instances defined in this class in a dictionary.
f_numeric(**kwargs)	Function call to update differential equation values.
g_numeric(**kwargs)	Function call to update algebraic equation values.
j_numeric(**kwargs)	Per-iteration numerical Jacobian update.
j_reset()	Helper function to clear the lists holding the numerical Jacobians.
j_setup()	One-time Jacobian sparsity pattern and constant value setup.

Attributes

class_name	Return the class name.

andes.core.block.PITrackAW

class andes.core.block.**PITrackAW**(*u*, *kp*, *ki*, *ks*, *lower*, *upper*, *no_lower=False*, *no_upper=False*, *ref=0.0*, *x0=0.0*, *name=None*, *tex_name=None*, *info=None*)

PI with tracking anti-windup limiter

__init__(*u*, *kp*, *ki*, *ks*, *lower*, *upper*, *no_lower=False*, *no_upper=False*, *ref=0.0*, *x0=0.0*, *name=None*, *tex_name=None*, *info=None*)

Methods

define()	Function for setting the initialization and equation strings for internal variables.
enforce_tex_name(fields)	Enforce tex_name is not None
export()	Method for exporting instances defined in this class in a dictionary.
f_numeric(**kwargs)	Function call to update differential equation values.
g_numeric(**kwargs)	Function call to update algebraic equation values.
j_numeric(**kwargs)	Per-iteration numerical Jacobian update.
j_reset()	Helper function to clear the lists holding the numerical Jacobians.
j_setup()	One-time Jacobian sparsity pattern and constant value setup.

Attributes

class_name	Return the class name.

andes.core.block.PIFreeze

class andes.core.block.**PIFreeze**(*u*, *kp*, *ki*, *freeze*, *ref=0.0*, *x0=0.0*, *name=None*, *tex_name=None*, *info=None*)

PI controller with state freeze.

Freezes state when the corresponding *freeze == 1*.

Notes

Tested in *experimental.TestPITrackAW.PIFreeze*.

__init__(*u*, *kp*, *ki*, *freeze*, *ref=0.0*, *x0=0.0*, *name=None*, *tex_name=None*, *info=None*)

Methods

`define()`	
	Notes
`enforce_tex_name`(fields)	Enforce tex_name is not None
`export()`	Method for exporting instances defined in this class in a dictionary.
`f_numeric`(**kwargs)	Function call to update differential equation values.
`g_numeric`(**kwargs)	Function call to update algebraic equation values.
`j_numeric`(**kwargs)	Per-iteration numerical Jacobian update.
`j_reset()`	Helper function to clear the lists holding the numerical Jacobians.
`j_setup()`	One-time Jacobian sparsity pattern and constant value setup.

Attributes

`class_name`	Return the class name.

andes.core.block.PITrackAWFreeze

class andes.core.block.**PITrackAWFreeze**(*u*, *kp*, *ki*, *ks*, *lower*, *upper*, *freeze*, *no_lower=False*, *no_upper=False*, *ref=0.0*, *x0=0.0*, *name=None*, *tex_name=None*, *info=None*)

PI controller with tracking anti-windup limiter and state freeze.

__init__(*u*, *kp*, *ki*, *ks*, *lower*, *upper*, *freeze*, *no_lower=False*, *no_upper=False*, *ref=0.0*, *x0=0.0*, *name=None*, *tex_name=None*, *info=None*)

Methods

`define()`	Function for setting the initialization and equation strings for internal variables.
`enforce_tex_name`(fields)	Enforce tex_name is not None
`export()`	Method for exporting instances defined in this class in a dictionary.
`f_numeric`(**kwargs)	Function call to update differential equation values.
`g_numeric`(**kwargs)	Function call to update algebraic equation values.
`j_numeric`(**kwargs)	Per-iteration numerical Jacobian update.
`j_reset()`	Helper function to clear the lists holding the numerical Jacobians.
`j_setup()`	One-time Jacobian sparsity pattern and constant value setup.

Attributes

`class_name`	Return the class name.

andes.core.block.PIDController

class `andes.core.block.`**`PIDController`**(*u*, *kp*, *ki*, *kd*, *Td*, *name*, *ref=0.0*, *x0=0.0*, *tex_name=None*, *info=None*)

Proportional Integral Derivative Controller.

```
     ┌──────────────────────┐
     │       ki      skd    │
u -> │kp + ─── + ───────    │ -> y
     │       s     1 + sTd  │
     └──────────────────────┘
```

The controller takes an error signal as the input. It takes an optional `ref` signal, which will be subtracted from the input.

The name is suggessted to be specified the same as the instance name.

This block assembles a `PIController` and a `Washout`.

Parameters

u
[BaseVar] The input variable instance

kp
[BaseParam] The proportional gain parameter instance

ki
[BaseParam] The integral gain parameter instance

kd
[BaseParam] The derivative gain parameter instance

Td
[BaseParam] The derivative time constant parameter instance

x0
[BaseParam] The initial value of the integrator

__init__(*u*, *kp*, *ki*, *kd*, *Td*, *name*, *ref=0.0*, *x0=0.0*, *tex_name=None*, *info=None*)

Methods

`define()`	Define equations for the PID Controller.
`enforce_tex_name(fields)`	Enforce tex_name is not None
`export()`	Method for exporting instances defined in this class in a dictionary.
`f_numeric(**kwargs)`	Function call to update differential equation values.
`g_numeric(**kwargs)`	Function call to update algebraic equation values.
`j_numeric(**kwargs)`	Per-iteration numerical Jacobian update.
`j_reset()`	Helper function to clear the lists holding the numerical Jacobians.
`j_setup()`	One-time Jacobian sparsity pattern and constant value setup.

Attributes

`class_name`	Return the class name.

andes.core.block.PIDAWHardLimit

class andes.core.block.**PIDAWHardLimit**(*u*, *kp*, *ki*, *kd*, *Td*, *aw_lower*, *aw_upper*, *lower*, *upper*, *name*, *no_lower=False*, *no_upper=False*, *ref=0.0*, *x0=0.0*, *tex_name=None*, *info=None*)

PID controller with anti-windup limiter on the integrator and hard limit on the output.

```
                  upper
                /------
     ┌──────────────────────┐
     │       ki      skd    │
```

(continues on next page)

(continued from previous page)

```
u -> |kp + ─── + ────────  | -> y
     |      s    1 + sTd  |
     └────────────────────┘

         ______/
         lower
```

The controller takes an error signal as the input.

Limits `lower` and `upper` are on the final output, and `aw_lower` `aw_upper` are on the integrator.

The name is suggessted to be specified the same as the instance name.

Parameters

u
[BaseVar] The input variable instance

kp
[BaseParam] The proportional gain parameter instance

ki
[BaseParam] The integral gain parameter instance

kd
[BaseParam] The derivative gain parameter instance

Td
[BaseParam] The derivative time constant parameter instance

__init__(*u*, *kp*, *ki*, *kd*, *Td*, *aw_lower*, *aw_upper*, *lower*, *upper*, *name*, *no_lower=False*, *no_upper=False*, *ref=0.0*, *x0=0.0*, *tex_name=None*, *info=None*)

Methods

`define`()	Define equations for the PI Controller.
`enforce_tex_name`(fields)	Enforce tex_name is not None
`export`()	Method for exporting instances defined in this class in a dictionary.
`f_numeric`(**kwargs)	Function call to update differential equation values.
`g_numeric`(**kwargs)	Function call to update algebraic equation values.
`j_numeric`(**kwargs)	Per-iteration numerical Jacobian update.
`j_reset`()	Helper function to clear the lists holding the numerical Jacobians.
`j_setup`()	One-time Jacobian sparsity pattern and constant value setup.

Attributes

class_name	Return the class name.

andes.core.block.PIDTrackAW

class andes.core.block.**PIDTrackAW**(*u*, *kp*, *ki*, *kd*, *Td*, *ks*, *lower*, *upper*, *no_lower=False*, *no_upper=False*, *ref=0.0*, *x0=0.0*, *name=None*, *tex_name=None*, *info=None*)

PID with tracking anti-windup limiter

__init__(*u*, *kp*, *ki*, *kd*, *Td*, *ks*, *lower*, *upper*, *no_lower=False*, *no_upper=False*, *ref=0.0*, *x0=0.0*, *name=None*, *tex_name=None*, *info=None*)

Methods

define()	Function for setting the initialization and equation strings for internal variables.
enforce_tex_name(fields)	Enforce tex_name is not None
export()	Method for exporting instances defined in this class in a dictionary.
f_numeric(**kwargs)	Function call to update differential equation values.
g_numeric(**kwargs)	Function call to update algebraic equation values.
j_numeric(**kwargs)	Per-iteration numerical Jacobian update.
j_reset()	Helper function to clear the lists holding the numerical Jacobians.
j_setup()	One-time Jacobian sparsity pattern and constant value setup.

Attributes

class_name	Return the class name.

Linear Blocks

class andes.core.block.**Gain**(*u*, *K*, *name=None*, *tex_name=None*, *info=None*)

Gain block.

```
     ┌───┐
u -> │ K │ -> y
     └───┘
```

Exports an algebraic output *y*.

define()

Implemented equation and the initial condition are

$$y = Ku$$
$$y^{(0)} = Ku^{(0)}$$

class andes.core.block.**GainLimiter**(*u*, *K*, *R*, *lower*, *upper*, *no_lower=False*, *no_upper=False*, *sign_lower=1*, *sign_upper=1*, *allow_adjust=True*, *name=None*, *tex_name=None*, *info=None*)

Gain followed by a limiter and another gain.

Exports the limited output *y*, unlimited output *x*, and HardLimiter *lim*.

```
     ┌─────┐          upper  ┌─────┐
     │     │         /-----  │     │
u -> │  K  │ -> x   /   ->   │  R  │ -> y
     │     │ _____/          │     │
     └─────┘ lower           └─────┘
```

Parameters

u
[str, BaseVar] Input variable, or an equation string for constructing an anonymous variable

K
[str, BaseParam, BaseService] Initial gain for *u* before limiter

R
[str, BaseParam, BaseService] Post limiter gain

define()

TODO: write docstring

class andes.core.block.**Piecewise**(*u*, *points: List | Tuple*, *funs: List | Tuple*, *name=None*, *tex_name=None*, *info=None*)

Piecewise block. Outputs an algebraic variable *y*.

This block takes a list of *N* points, *[x0, x1, ...x_{n-1}]* to define N+1 ranges, namely (-inf, x0), (x0, x1), ..., (x_{n-1}, +inf). and a list of *N+1* function strings *[fun0, ..., fun_n]*.

Inputs that fall within each range applies the corresponding function. The first range (-inf, x0) applies *fun_0*, and the last range (x_{n-1}, +inf) applies the last function *fun_n*.

The function returns zero if no condition is met.

Note: Piecewise.y must remain `Algeb` (not `Observable`) because: (1) model authors may set `v_iter` on the output for coupled iterative initialization (e.g., EXAC1), and (2) zero-valued branches cause `ComplexInfinity` when the expression is substituted into denominators during code generation.

Parameters

points
[list, tuple] A list of piecewise points. Need to be provided in the constructor function.

funs
[list, tuple] A list of strings for the piecewise functions. Need to be provided in the overloaded *define* function.

define()

Build the equation string for the piecewise equations.

`self.funs` needs to be provided with the function strings corresponding to each range.

class andes.core.block.**HVGate**(*u1*, *u2*, *name=None*, *tex_name=None*, *info=None*)

High Value Gate. Outputs the maximum of two inputs.

```
      ┌──────────┐
u1 -> │ HV Gate  │
      │          │ ->  y
u2 -> │   (MAX)  │
      └──────────┘
```

class andes.core.block.**LVGate**(*u1*, *u2*, *name=None*, *tex_name=None*, *info=None*)

Low Value Gate. Outputs the minimum of the two inputs.

```
      ┌──────────┐
u1 -> │ LV Gate  │
      │          │ ->  y
u2 -> │   (MIN)  │
      └──────────┘
```

class andes.core.block.**DeadBand1**(*u*, *center*, *lower*, *upper*, *gain=1.0*, *enable=True*, *name=None*, *tex_name=None*, *info=None*, *namespace='local'*)

Deadband type 1 (linear, non-step).

Note: DeadBand1.y must remain `Algeb` (not `Observable`) because other models (e.g., DGPRCT1)

reference it via `ExtAlgeb`.

Parameters

center
Default value when within the deadband. If the input is an error signal, center should be set to zero.

gain
Gain multiplied to DeadBand discrete block's output.

Notes

Block diagram

```
      |   /
______|__/___   -> Gain -> DeadBand1_y
   /  |
  /   |
```

First Order Blocks

class andes.core.block.**Integrator**(*u*, *T*, *K*, *y0*, *check_init=True*, *name=None*, *tex_name=None*, *info=None*)

Integrator block.

```
u -> ┌─────┐
     │ K/sT │ -> y
     └─────┘
```

Exports a differential variable *y*.

The initial output needs to be specified through *y0*.

define()

Implemented equation and the initial condition are

$$\dot{y} = Ku$$
$$y^{(0)} = 0$$

class andes.core.block.**IntegratorAntiWindup**(*u*, *T*, *K*, *y0*, *lower*, *upper*, *name=None*, *tex_name=None*, *info=None*, *no_warn=False*)

Integrator block with anti-windup limiter.

```
           upper
          /-----
      ┌─────────┐
u ->  │  K/sT   │  -> y
      └─────────┘

   _____/
   lower
```

Exports a differential variable *y* and an AntiWindup *lim*. The initial output must be specified through *y0*.

define()

Implemented equation and the initial condition are

$$\dot{y} = Ku$$
$$y^{(0)} = 0$$

class andes.core.block.**Lag**(*u*, *T*, *K*, *D=1*, *name=None*, *tex_name=None*, *info=None*)

Lag (low pass filter) transfer function.

```
      ┌─────────┐
      │    K    │
u ->  │ ─────── │  -> y
      │ D + sT  │
      └─────────┘
```

Exports one state variable *y* as the output.

Parameters

K
Gain

T
Time constant

D
Constant

u
Input variable

define()

Notes

Equations and initial values are

$$T\dot{y} = (Ku - Dy)$$
$$y^{(0)} = Ku/D$$

class andes.core.block.**LagAntiWindup**(*u*, *T*, *K*, *lower*, *upper*, *D=1*, *name=None*, *tex_name=None*, *info=None*)

Lag (low pass filter) transfer function block with an anti-windup limiter.

```
              upper
            /------
      ┌──────────┐
      │    K     │
u ->  │ ──────── │ -> y
      │ D + sT   │
      └──────────┘
   ______/
   lower
```

Exports one state variable *y* as the output and one AntiWindup instance *lim*.

Parameters

K
Gain

T
Time constant

D
Constant

u
Input variable

define()

Notes

Equations and initial values are

$$T\dot{y} = (Ku - Dy)$$
$$y^{(0)} = Ku/D$$

class andes.core.block.**LagFreeze**(*u*, *T*, *K*, *freeze*, *D=1*, *name=None*, *tex_name=None*, *info=None*)

Lag with an input to freeze the state.

During the period when the freeze signal is 1, the LagFreeze output will be frozen.

define()

Notes

Equations and initial values are

$$T\dot{y} = (1 - freeze) * (Ku - y)$$
$$y^{(0)} = Ku$$

class andes.core.block.**LagAWFreeze**(*u*, *T*, *K*, *lower*, *upper*, *freeze*, *D=1*, *name=None*, *tex_name=None*, *info=None*)

Lag with anti-windup limiter and state freeze.

Note that the output *y* is a state variable.

define()

Notes

Equations and initial values are

$$T\dot{y} = (1 - freeze)(Ku - y)$$
$$y^{(0)} = Ku$$

`y` undergoes an anti-windup limiter.

class andes.core.block.**LagRate**(*u*, *T*, *K*, *rate_lower*, *rate_upper*, *D=1*, *rate_no_lower=False*, *rate_no_upper=False*, *rate_lower_cond=None*, *rate_upper_cond=None*, *name=None*, *tex_name=None*, *info=None*)

Lag (low pass filter) transfer function block with a rate limiter.

```
              / rate_upper
        ┌─────────┐
        │    K    │
  u ->  │ ─────── │ -> y
        │ D + sT  │
        └─────────┘
rate_lower /
```

Exports one state variable *y* as the output and one AntiWindupRate instance *lim*.

Parameters

K
Gain

T
Time constant

D
Constant

u
Input variable

define()

Notes

Equations and initial values are

$$T\dot{y} = (Ku - y)$$
$$y^{(0)} = Ku$$

class andes.core.block.**LagAntiWindupRate**(*u*, *T*, *K*, *lower*, *upper*, *rate_lower*, *rate_upper*, *D=1*, *no_lower=False*, *no_upper=False*, *rate_no_lower=False*, *rate_no_upper=False*, *rate_lower_cond=None*, *rate_upper_cond=None*, *name=None*, *tex_name=None*, *info=None*)

Lag (low pass filter) transfer function block with a rate limiter and an anti-windup limiter.

```
             upper
rate_upper /------
     ┌──────────┐
     │    K     │
u -> │ ──────── │ -> y
     │ D + sT   │
     └──────────┘
   ______/ rate_lower
   lower
```

Exports one state variable *y* as the output and one AntiWindupRate instance *lim*.

Parameters

K
Gain

T
Time constant

D
Constant

u
Input variable

define()

Notes

Equations and initial values are

$$T\dot{y} = (Ku - Dy)$$
$$y^{(0)} = Ku/D$$

class andes.core.block.**Washout**(*u*, *T*, *K*, *name=None*, *tex_name=None*, *info=None*)

Washout filter (high pass) block.

```
      ┌─────────┐
      │   sK    │
u ->  │ ─────── │ -> y
      │ 1 + sT  │
      └─────────┘
```

Exports state x (symbol x') and output algebraic variable y.

define()

Notes

Equations and initial values:

$$T\dot{x'} = (u - x')$$
$$Ty = K(u - x')$$
$$x'^{(0)} = u$$
$$y^{(0)} = 0$$

class andes.core.block.**WashoutOrLag**(*u*, *T*, *K*, *name=None*, *zero_out=True*, *tex_name=None*, *info=None*)

Washout with the capability to convert to Lag when K = 0.

Can be enabled with *zero_out*. Need to provide *name* to construct.

Exports state x (symbol x'), output algebraic variable y, and a LessThan block *LT*.

Parameters

zero_out
[bool, optional] If True, `sT` will become 1, and the washout will become a low-pass filter. If False, functions as a regular Washout.

define()

Notes

Equations and initial values:

$$
\begin{aligned}
T\dot{x'} &= (u - x') \\
Ty = z_0 K(u - x') + z_1 T x & \\
x'^{(0)} &= u \\
y^{(0)} &= 0
\end{aligned}
$$

where `z_0` is a flag array for the greater-than-zero elements, and `z_1` is that for the less-than or equal-to zero elements.

class `andes.core.block.`**`LeadLag`**(*u*, *T1*, *T2*, *K=1*, *zero_out=True*, *name=None*, *tex_name=None*, *info=None*)

Lead-Lag transfer function block in series implementation.

```
      ┌─────────────┐
      │   1 + sT1   │
u ->  │ K ───────── │ -> y
      │   1 + sT2   │
      └─────────────┘
```

Exports two variables: internal state `x` and output algebraic variable `y`.

Parameters

u
[BaseVar or BaseParam] Input signal.

T1
[BaseParam] Numerator (lead) time constant.

T2
[BaseParam] Denominator (lag) time constant. Also used as the `t_const` of the internal state `x`.

K
[BaseParam or numeric, optional] Static gain (default 1).

zero_out
[bool, optional] If `True` (default), bypass the block as `y = K * u` when T2 <= 0. Set to `False` only if T2 > 0 is guaranteed by the data.

define()

Notes

Implemented equations and initial values

$$
\begin{aligned}
T_2\dot{x'} &= (u - x') \\
T_2 y &= KT_1(u - x') + KT_2 x' + E_2 \text{ , where} \\
E_2 &= \begin{cases} (y - Ku) & \text{if } T_2 \leq 0 \& zero_out = True \\ 0 & \text{otherwise} \end{cases} \\
x'^{(0)} &= u \\
y^{(0)} &= Ku
\end{aligned}
$$

class andes.core.block.**LeadLagLimit**(*u*, *T1*, *T2*, *lower*, *upper*, *name=None*, *tex_name=None*, *info=None*)

Lead-Lag transfer function block with hard limiter (series implementation).

```
      ┌──────────┐           upper
      │ 1 + sT1  │          /-----
u ->  │ ──────── │ -> ynl  / -> y
      │ 1 + sT2  │  ______/
      └──────────┘  lower
```

Exports four variables: state *x*, output before hard limiter *ynl*, output *y*, and AntiWindup *lim*.

define()

Notes

Implemented control block equations (without limiter) and initial values

$$
\begin{aligned}
T_2\dot{x'} &= (u - x') \\
T_2 y &= T_1(u - x') + T_2 x' \\
x'^{(0)} &= y^{(0)} = u
\end{aligned}
$$

Second Order Blocks

class andes.core.block.**Lag2ndOrd**(*u*, *K*, *T1*, *T2*, *name=None*, *tex_name=None*, *info=None*)

Second order lag transfer function (low-pass filter).

```
      ┌──────────────────┐
      │        K         │
u ->  │ ──────────────── │ -> y
      │ 1 + sT1 + s^2 T2 │
      └──────────────────┘
```

Exports one two state variables (*x*, *y*), where *y* is the output.

Parameters

u
Input

K
Gain

T1
First order time constant

T2
Second order time constant

define()

Notes

Implemented equations and initial values are

$$
\begin{aligned}
T_2\dot{x} &= Ku - y - T_1 x \\
\dot{y} &= x \\
x^{(0)} &= 0 \\
y^{(0)} &= Ku
\end{aligned}
$$

class andes.core.block.**LeadLag2ndOrd**(*u*, *T1*, *T2*, *T3*, *T4*, *zero_out=False*, *name=None*, *tex_name=None*, *info=None*)

Second-order lead-lag transfer function block.

```
      ┌─────────────────────┐
      │ 1 + sT3 + s^2 T4 │
u ->  │ ─────────────────── │ -> y
      │ 1 + sT1 + s^2 T2 │
      └─────────────────────┘
```

Exports two internal states (`x1` and `x2`) and output algebraic variable `y`.

Parameters

u
[BaseVar or BaseParam] Input signal.

T1
[BaseParam] Denominator first-order time constant.

T2
[BaseParam] Denominator second-order time constant. Also used as the `t_const` of the internal state `x1`.

T3
[BaseParam] Numerator first-order time constant.

T4
[BaseParam] Numerator second-order time constant.

zero_out
[bool, optional] If `True`, bypass the block as `y = u` when T2 <= 0. Default is `False`.

define()

Notes

Implemented equations and initial values are

$$
\begin{aligned}
T_2\dot{x}_1 &= u - x_2 - T_1 x_1 \\
\dot{x}_2 &= x_1 \\
T_2 y &= T_2 x_2 + T_2 T_3 x_1 + T_4(u - x_2 - T_1 x_1) + E_2\,, \text{ where} \\
E_2 &= \begin{cases} (y-u) & \text{if } T_2 \le 0 \& zero_out = True \\ 0 & \text{otherwise} \end{cases} \\
x_1^{(0)} &= 0 \\
x_2^{(0)} &= y^{(0)} = u
\end{aligned}
$$

PI Controllers

class andes.core.block.**PIController**(*u*, *kp*, *ki*, *ref=0.0*, *x0=0.0*, *name=None*, *tex_name=None*, *info=None*)

Proportional Integral Controller.

The controller takes an error signal as the input. It takes an optional *ref* signal, which will be subtracted from the input.

Parameters

u
[BaseVar] The input variable instance

kp
[BaseParam] The proportional gain parameter instance

ki
[[type]] The integral gain parameter instance

define()

Define equations for the PI Controller.

Notes

One state variable `xi` and one algebraic variable `y` are added.

Equations implemented are

$$\dot{x_i} = k_i * (u - ref)$$
$$y = x_i + k_p * (u - ref)$$

class andes.core.block.**PIAWHardLimit**(*u*, *kp*, *ki*, *aw_lower*, *aw_upper*, *lower*, *upper*, *no_lower=False*, *no_upper=False*, *ref=0.0*, *x0=0.0*, *name=None*, *tex_name=None*, *info=None*)

PI controller with anti-windup limiter on the integrator and hard limit on the output.

Limits `lower` and `upper` are on the final output, and `aw_lower` `aw_upper` are on the integrator.

define()

Define equations for the PI Controller.

Notes

One state variable `xi` and one algebraic variable `y` are added.

Equations implemented are

$$\dot{x_i} = k_i * (u - ref)$$
$$y = x_i + k_p * (u - ref)$$

class andes.core.block.**PITrackAW**(*u*, *kp*, *ki*, *ks*, *lower*, *upper*, *no_lower=False*, *no_upper=False*, *ref=0.0*, *x0=0.0*, *name=None*, *tex_name=None*, *info=None*)

PI with tracking anti-windup limiter

define()

Function for setting the initialization and equation strings for internal variables. This method must be implemented by subclasses.

The equations should be written with the "final" variable names. Let's say the block instance is named *blk* (kept at `self.name` of the block), and an internal variable *v* is defined. The internal variable will be captured as `blk_v` by the parent model. Therefore, all equations should use `{self.name}_v` to represent variable `v`, where `{self.name}` is the name of the block at run time.

On the other hand, the names of externally provided parameters or variables are obtained by directly accessing the `name` attribute. For example, if `self.T` is a parameter provided through the block constructor, `{self.T.name}` should be used in the equation.

See also:

PIController.define

Equations for the PI Controller block

Examples

An internal variable `v` has a trivial equation `T = v`, where T is a parameter provided to the block constructor.

In the model, one has

```
class SomeModel():
    def __init__(...)
        self.input = Algeb()
        self.T = Param()

        self.blk = ExampleBlock(u=self.input, T=self.T)
```

In the ExampleBlock function, the internal variable is defined in the constructor as

```
class ExampleBlock():
    def __init__(...):
        self.v = Algeb()
        self.vars = {'v', self.v}
```

In the `define`, the equation is provided as

```
def define(self):
    self.v.v_str = '{self.T.name}'
    self.v.e_str = '{self.T.name} - {self.name}_v'
```

In the parent model, `v` from the block will be captured as `blk_v`, and the equation will evaluate into

```
self.blk_v.v_str = 'T'
self.blk_v.e_str = 'T - blk_v'
```

class andes.core.block.**PIFreeze**(*u*, *kp*, *ki*, *freeze*, *ref=0.0*, *x0=0.0*, *name=None*, *tex_name=None*, *info=None*)

PI controller with state freeze.

Freezes state when the corresponding *freeze == 1*.

Notes

Tested in *experimental.TestPITrackAW.PIFreeze*.

define()

Notes

One state variable `xi` and one algebraic variable `y` are added.

Equations implemented are

$$\dot{x_i} = k_i * (u - ref)$$
$$y = (1 - freeze) * (x_i + k_p * (u - ref)) + freeze * y$$

class andes.core.block.**PITrackAWFreeze**(*u*, *kp*, *ki*, *ks*, *lower*, *upper*, *freeze*, *no_lower=False*, *no_upper=False*, *ref=0.0*, *x0=0.0*, *name=None*, *tex_name=None*, *info=None*)

PI controller with tracking anti-windup limiter and state freeze.

define()

Function for setting the initialization and equation strings for internal variables. This method must be implemented by subclasses.

The equations should be written with the "final" variable names. Let's say the block instance is named *blk* (kept at `self.name` of the block), and an internal variable *v* is defined. The internal variable will be captured as `blk_v` by the parent model. Therefore, all equations should use `{self.name}_v` to represent variable `v`, where `{self.name}` is the name of the block at run time.

On the other hand, the names of externally provided parameters or variables are obtained by directly accessing the `name` attribute. For example, if `self.T` is a parameter provided through the block constructor, `{self.T.name}` should be used in the equation.

See also:

PIController.define
Equations for the PI Controller block

Examples

An internal variable `v` has a trivial equation `T = v`, where T is a parameter provided to the block constructor.

In the model, one has

```
class SomeModel():
    def __init__(...)
        self.input = Algeb()
        self.T = Param()

        self.blk = ExampleBlock(u=self.input, T=self.T)
```

In the ExampleBlock function, the internal variable is defined in the constructor as

```
class ExampleBlock():
    def __init__(...):
        self.v = Algeb()
        self.vars = {'v', self.v}
```

In the `define`, the equation is provided as

```
def define(self):
    self.v.v_str = '{self.T.name}'
    self.v.e_str = '{self.T.name} - {self.name}_v'
```

In the parent model, `v` from the block will be captured as `blk_v`, and the equation will evaluate into

```
self.blk_v.v_str = 'T'
self.blk_v.e_str = 'T - blk_v'
```

class andes.core.block.**PIDController**(*u*, *kp*, *ki*, *kd*, *Td*, *name*, *ref=0.0*, *x0=0.0*, *tex_name=None*, *info=None*)

Proportional Integral Derivative Controller.

```
      ┌──────────────────────┐
      │       ki      skd    │
u ->  │kp + ─── + ────────   │ -> y
      │       s     1 + sTd  │
      └──────────────────────┘
```

The controller takes an error signal as the input. It takes an optional `ref` signal, which will be subtracted from the input.

The name is suggessted to be specified the same as the instance name.

This block assembles a `PIController` and a `Washout`.

Parameters

u
[BaseVar] The input variable instance

kp
[BaseParam] The proportional gain parameter instance

ki
[BaseParam] The integral gain parameter instance

kd
[BaseParam] The derivative gain parameter instance

Td
[BaseParam] The derivative time constant parameter instance

x0
[BaseParam] The initial value of the integrator

define()

Define equations for the PID Controller.

Notes

One PIController `PIC`, one Washout `xd`, and one algebraic variable `y` are added.

Equations implemented are

$$\dot{x_i} = k_i * (u - ref)$$
$$xd = Washout(u - ref)$$
$$y = x_i + k_p * (u - ref) + xd$$

class andes.core.block.**PIDAWHardLimit**(*u*, *kp*, *ki*, *kd*, *Td*, *aw_lower*, *aw_upper*, *lower*, *upper*, *name*, *no_lower=False*, *no_upper=False*, *ref=0.0*, *x0=0.0*, *tex_name=None*, *info=None*)

PID controller with anti-windup limiter on the integrator and hard limit on the output.

```
                  upper
                /------
      ┌──────────────────────┐
      │       ki     skd     │
u ->  │kp + ─── + ────────   │ -> y
      │       s    1 + sTd   │
      └──────────────────────┘

          ______/
          lower
```

The controller takes an error signal as the input.

Limits `lower` and `upper` are on the final output, and `aw_lower` `aw_upper` are on the integrator.

The name is suggessted to be specified the same as the instance name.

Parameters

u
[BaseVar] The input variable instance

kp
[BaseParam] The proportional gain parameter instance

ki
[BaseParam] The integral gain parameter instance

kd
[BaseParam] The derivative gain parameter instance

Td
[BaseParam] The derivative time constant parameter instance

define()

Define equations for the PI Controller.

Notes

One state variable `xi` and one algebraic variable `y` are added.

Equations implemented are

$$\dot{x_i} = k_i * (u - ref)$$
$$y = x_i + k_p * (u - ref)$$

class andes.core.block.**PIDTrackAW**(*u*, *kp*, *ki*, *kd*, *Td*, *ks*, *lower*, *upper*, *no_lower=False*, *no_upper=False*, *ref=0.0*, *x0=0.0*, *name=None*, *tex_name=None*, *info=None*)

PID with tracking anti-windup limiter

define()

Function for setting the initialization and equation strings for internal variables. This method must be implemented by subclasses.

The equations should be written with the "final" variable names. Let's say the block instance is named *blk* (kept at `self.name` of the block), and an internal variable *v* is defined. The internal variable will be captured as `blk_v` by the parent model. Therefore, all equations should use `{self.name}_v` to represent variable `v`, where `{self.name}` is the name of the block at run time.

On the other hand, the names of externally provided parameters or variables are obtained by directly accessing the `name` attribute. For example, if `self.T` is a parameter provided through the block constructor, `{self.T.name}` should be used in the equation.

See also:

PIController.define
Equations for the PI Controller block

Examples

An internal variable `v` has a trivial equation `T = v`, where T is a parameter provided to the block constructor.

In the model, one has

```
class SomeModel():
    def __init__(...)
        self.input = Algeb()
        self.T = Param()

        self.blk = ExampleBlock(u=self.input, T=self.T)
```

In the ExampleBlock function, the internal variable is defined in the constructor as

```
class ExampleBlock():
    def __init__(...):
        self.v = Algeb()
        self.vars = {'v', self.v}
```

In the `define`, the equation is provided as

```
def define(self):
    self.v.v_str = '{self.T.name}'
    self.v.e_str = '{self.T.name} - {self.name}_v'
```

In the parent model, `v` from the block will be captured as `blk_v`, and the equation will evaluate into

```
self.blk_v.v_str = 'T'
self.blk_v.e_str = 'T - blk_v'
```

Saturation

class andes.models.exciter.**ExcExpSat**(*E1*, *SE1*, *E2*, *SE2*, *name=None*, *tex_name=None*, *info=None*)

Exponential exciter saturation block to calculate A and B from E1, SE1, E2 and SE2. Input parameters will be corrected and the user will be warned. To disable saturation, set either E1 or E2 to 0.

Parameters

E1
[BaseParam] First point of excitation field voltage

SE1: BaseParam
Coefficient corresponding to E1

E2
[BaseParam] Second point of excitation field voltage

SE2: BaseParam
Coefficient corresponding to E2

define()

Notes

The implementation solves for coefficients *A* and *B* which satisfy

$$E_1 S_{E1} = A e^{E1 \times B} E_2 S_{E2} = A e^{E2 \times B}$$

The solutions are given by

$$E_1 S_{E1} e^{\frac{E_1 \log\left(\frac{E_2 S_{E2}}{E_1 S_{E1}}\right)}{E_1 - E_2}} - \frac{\log\left(\frac{E_2 S_{E2}}{E_1 S_{E1}}\right)}{E_1 - E_2}$$

Usage Examples

First-Order Lag

```
from andes.core.block import Lag

self.LG = Lag(
    u=self.x,    # Input
    T=self.T1,   # Time constant
    K=1          # Gain (default=1)
)
# Output: LG_y
```

Washout Filter

```
from andes.core.block import Washout

self.WO = Washout(
    u=self.x,
    T=self.Tw,
    K=self.Kw
)
# Output: WO_y
```

Lead-Lag Compensator

```
from andes.core.block import LeadLag

self.LL = LeadLag(
    u=self.x,
    T1=self.T1,   # Lead time constant
    T2=self.T2    # Lag time constant
)
# Output: LL_y
```

PI Controller with Anti-Windup

```
from andes.core.block import PIAWHardLimit

self.PIAW = PIAWHardLimit(
    u=self.error,
    kp=self.Kp,
    ki=self.Ki,
    lower=0,
    upper=1
```

(continues on next page)

(continued from previous page)

```
)
# Output: PIAW_y
```

Chaining Blocks

Blocks can be chained by using one block's output as another's input:

```
# Input → Lag → LeadLag → Output
self.LG = Lag(u=self.input, T=self.T1, K=1)
self.LL = LeadLag(u=self.LG_y, T1=self.T2, T2=self.T3)
self.output = Algeb(e_str='LL_y - output')
```

Naming Convention

We loosely follow a naming convention when using modeling blocks. An instance of a modeling block is named with a two-letter acronym, followed by a number or a meaningful but short variable name. The acronym and the name are spelled in one word without underscore, as the output of the block already contains `_y`.

For example, two washout filters can be named `WO1` and `WO2`. In another case, a first-order lag function for voltage sensing can be called `LGv`, or even `LG` if there is only one Lag instance in the model.

Block	Acronym	Example
Lag	LG	LG, LGv, LG1
Washout	WO	WO, WO1, WO2
LeadLag	LL	LL, LL1, LLv
Integrator	IN	IN, INT
PIController	PI	PI, PIv

Naming conventions are not strictly enforced. Expressiveness and concision are encouraged.

See Also

- *Discrete Components* - Discrete components (many blocks use these internally)
- *Variables* - Variable types
- *Example: TGOV1 Turbine Governor* - TGOV1 example using blocks

4.3.6 Groups

A group is a collection of similar functional models with common variables and parameters. It is mandatory to enforce the common variables and parameters when developing new models. The common variables and parameters are typically the interface when connecting different group models.

Background

For example, the Group `RenGen` has variables `Pe` and `Qe`, which are active power output and reactive power output. Such common variables can be retrieved by other models, such as one in the Group `RenExciter` for further calculation.

In such a way, the same variable interface is realized so that all models in the same group can carry out similar functions. This enables:

1. **Common interface**: All generators have `bus`, `Sn`, `Vn`
2. **Polymorphism**: Exciters work with any synchronous generator model
3. **Backward references**: Find all devices connected to a bus
4. **Model substitution**: Replace GENCLS with GENROU without changing case files

GroupBase

GroupBase()	Base class for groups.

andes.models.group.GroupBase

class andes.models.group.**GroupBase**

Base class for groups.

__init__()

Methods

`add`(idx, model)	Register an idx from model_name to the group
`add_model`(name, instance)	Add a Model instance to group.
`alter`(src, idx, value[, attr])	Alter values of input parameters or constant service for a group of models.
`as_df`([vin])	Export group common parameters as a *pandas.DataFrame* object.
`as_dict`([vin])	Export group common parameters as a dictionary.
`doc`([export])	Return the documentation of the group in a string.
`doc_all`([export])	Return documentation of the group and its models.
`find_idx`(keys, values[, allow_none, ...])	Find indices of devices that satisfy the given *key=value* condition.
`get`(src, idx[, attr, allow_none, default])	Based on the indexer, get the *attr* field of the *src* parameter or variable.
`get_all_idxes`()	Return all the devices idx in this group.
`get_field`(src, idx, field)	Helper function for retrieving an attribute of a member variable shared by models in this group.
`get_next_idx`([idx, model_name])	Get a no-conflict idx for a new device.
`get_setpoint`(system, idx, name)	Get the current setpoint value by resolving the controller chain.
`get_status`(idx)	Get the effective online status of a device.
`idx2model`(idx[, allow_none])	Find model name for the given idx.
`idx2uid`(idx)	Convert idx to the 0-indexed unique index.
`set`(src, idx, *args[, value, attr, base])	Set the value of a group property, dispatching to the correct model.
`set_backref`(name, from_idx, to_idx)	Set idxes to `BackRef`, and set them to models.
`set_setpoint`(system, idx, name, value)	Set a setpoint value by resolving the controller chain.
`set_status`(idx, value)	Set the online status of a device and propagate to dependents.

Attributes

`class_name`	
`n`	Total number of devices.

class `andes.models.group.`**`GroupBase`**

Base class for groups.

add(*idx*, *model*)

Register an idx from model_name to the group

Parameters

idx: Union[str, float, int]
Register an element to a model

model: Model
instance of the model

Returns

add_model (*name: str*, *instance*)

Add a Model instance to group.

Parameters

name
[str] Model name

instance
[Model] Model instance

Returns

None

alter (*src*, *idx*, *value*, *attr='v'*)

Alter values of input parameters or constant service for a group of models.

Deprecated since version Use: `set()` with `base='device'` instead.

Parameters

src
[str] The parameter name to alter

idx
[str, float, int] The unique identifier for the device to alter

value
[float] The desired value

attr
[str, optional] The attribute to alter. Default is 'v'.

as_df (*vin=False*)

Export group common parameters as a *pandas.DataFrame* object.

Note: New in version 1.9.3.

Parameters

vin
[bool] If True, export all parameters from original input (`vin`).

Returns

DataFrame
A dataframe containing all model data. An *uid* column is added.

as_dict(*vin=False*)

Export group common parameters as a dictionary.

Note: New in version 1.9.3.

This method returns a dictionary where the keys are the *ModelData* parameter names and the values are array-like structures containing the data in the order they were added. Unlike *ModelData.as_dict()*, this dictionary does not include the *uid* field.

Parameters

vin
[bool, optional] If True, includes the *vin* attribute in the dictionary. Default is False.

Returns

dict
A dictionary of common parameters.

doc(*export='plain'*)

Return the documentation of the group in a string.

doc_all(*export='plain'*)

Return documentation of the group and its models.

Parameters

export
['plain' or 'rest'] Export format, plain-text or RestructuredText

Returns

str

find_idx(*keys*, *values*, *allow_none=False*, *default=None*, *allow_all=False*)

Find indices of devices that satisfy the given *key=value* condition.

This method iterates over all models in this group.

Parameters

keys
[str, array-like, Sized] A string or an array-like of strings containing the names of parameters for the search criteria.

values
[array, array of arrays, Sized] Values for the corresponding key to search for. If keys is a str, values should be an array of elements. If keys is a list, values should be an array of arrays, each corresponding to the key.

allow_none
[bool, optional] Allow key, value to be not found. Used by groups. Default is False.

default
[bool, optional] Default idx to return if not found (missing). Default is None.

allow_all
[bool, optional] Return all matches if set to True. Default is False.

Returns

list
Indices of devices.

get(*src: str*, *idx*, *attr: str = 'v'*, *allow_none=False*, *default=0.0*)

Based on the indexer, get the *attr* field of the *src* parameter or variable.

Parameters

src
[str] param or var name

idx
[array-like] device idx

attr
The attribute of the param or var to retrieve

allow_none
[bool] True to allow None values in the indexer

default
[float] If *allow_none* is true, the default value to use for None indexer.

Returns

The requested param or variable attribute. If *idx* is a list, return a list of values.
If *idx* is a single element, return a single value.

get_all_idxes()

Return all the devices idx in this group.

Note: New in version 1.9.3.

Returns

list
List of indices.

Notes

The default models sequence depends on the order of the models in the group, which comes from OrderedDict *file_classes* in *models.__init__.py*.

Examples

```
>>> ss = andes.load(andes.get_case('ieee14/ieee14_pvd1.xlsx'))
>>> ss.DG.get_all_idxes()
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

```
>>> ss.StaticGen.get_all_idxes()
[2, 3, 4, 5, 6, 1]
```

get_field(*src: str*, *idx*, *field: str*)

Helper function for retrieving an attribute of a member variable shared by models in this group.

Returns

list
A list with the length equal to `len(idx)`.

get_next_idx(*idx=None*, *model_name=None*)

Get a no-conflict idx for a new device. Use the provided `idx` if no conflict. Generate a new one otherwise.

Parameters

idx
[str or None] Proposed idx. If None, assign a new one.

model_name
[str or None] Model name. If not, prepend the group name.

Returns

str
New device name.

get_setpoint(*system*, *idx*, *name*)

Get the current setpoint value by resolving the controller chain.

See `set_setpoint()` for notes on the physical meaning of setpoint values.

Parameters

system
[System] The system instance.

idx
[str, int, float] Device idx in this group.

name
[str] Setpoint key (e.g., `'pref'`, `'vref'`, `'qref'`).

Returns

float
Current value of the setpoint (system-base per-unit).

get_status(*idx*)

Get the effective online status of a device.

Parameters

idx
[str, int, float] Device idx.

Returns

float
Effective status value (0 or 1).

idx2model(*idx*, *allow_none=False*)

Find model name for the given idx.

Parameters

idx
[float, int, str, array-like] idx or idx-es of devices.

allow_none
[bool] If True, return *None* at the positions where idx is not found.

Returns

If *idx* is a list, return a list of model instances.
If *idx* is a single element, return a model instance.

idx2uid(*idx*)

Convert idx to the 0-indexed unique index.

Parameters

idx
[array-like, numbers, or str] idx of devices

Returns

list
A list containing the unique indices of the devices

property n

Total number of devices.

set(*src: str*, *idx*, **args*, *value=<object object>*, *attr='v'*, *base=None*)

Set the value of a group property, dispatching to the correct model.

Delegates to each model's `Model.set()`. See its docstring for details on `base` and `attr`.

Parameters

src
[str] Name of property.

idx
[str, int, float, array-like] Indices of devices.

value
[float or array-like] New values to be set. Can be passed as the third positional argument or as a keyword.

attr
[str, optional] Attribute to write (default `'v'`). Ignored when `base='device'`.

base
[`None` or `'device'`, optional] `None` (default): system-base direct write. `'device'`: device/input-base with per-unit conversion.

Returns

bool
True when successful.

`set_backref`(*name*, *from_idx*, *to_idx*)

Set idxes to `BackRef`, and set them to models.

`set_setpoint`(*system*, *idx*, *name*, *value*)

Set a setpoint value by resolving the controller chain.

Searches for a connected controller (e.g., TurbineGov for `'pref'`, Exciter for `'vref'`) and writes there. Falls back to the device itself with a warning.

Note: The physical meaning of a setpoint depends on the specific controller model that owns it. For example, `'vref'` writes to the exciter's voltage reference input, which may include compensation terms (e.g., EXAC4 initializes `vref0 = v + vf0/KA`). It is **not** necessarily the exact terminal voltage. Similarly, `'pref'` is the governor's power reference in system-base per-unit. Use `get_setpoint()` to read the current value before applying incremental changes.

Parameters

system
[System] The system instance.

idx
[str, int, float] Device idx in this group.

name
[str] Setpoint key (e.g., `'pref'`, `'vref'`, `'qref'`).

value
[float] Value to write (absolute, in system-base per-unit).

set_status(*idx*, *value*)

Set the online status of a device and propagate to dependents.

Delegates to the concrete model's `set_status()` which calls `System.set_status()` for propagation.

Parameters

idx
[str, int, float] Device idx.

value
[int or float] New status value (0 or 1).

Standard Groups

Group	Models	Common Parameters
`StaticGen`	PV, Slack	bus, p0, q0, Sn
`SynGen`	GENCLS, GENROU, GENSAL	bus, gen, Sn, Vn
`Exciter`	ESST1A, ESDC2, SEXS	syn
`TurbineGov`	TGOV1, IEEEG1, HYGOV	syn
`PSS`	IEEEST, ST2CUT	avr
`StaticLoad`	PQ, ZIP	bus, Vn
`Motor`	MotorThree, MotorFive	bus
`RenGen`	REGCA1, REECA1	bus, gen

Defining Groups

Groups are defined in `andes/models/group.py`:

```
from andes.core.model import GroupBase

class Exciter(GroupBase):
    """Exciter group base class."""

    def __init__(self):
        super().__init__()
        # Common parameters for all exciters
        self.common_params = ['syn']
        self.common_vars = []
```

Model Group Registration

Models declare their group in the constructor:

```
class ESST1A(ExciterData, Model):
    def __init__(self, system, config):
        ExciterData.__init__(self)
        Model.__init__(self, system, config)

        self.group = 'Exciter'  # Register with Exciter group
```

Using Groups for References

External Parameters

Get parameters from any model in a group:

```
class TurbineGov(Model):
    def __init__(self):
        # Reference generator - can be any SynGen model
        self.syn = IdxParam(model='SynGen', mandatory=True)

        # Get inertia from the generator (works for GENCLS or GENROU)
        self.M = ExtParam(src='M', model='SynGen', indexer=self.syn)
```

External Variables

Access variables from any group member:

```
class Exciter(Model):
    def __init__(self):
        self.syn = IdxParam(model='SynGen')

        # Access generator terminal voltage
        self.vd = ExtAlgeb(src='vd', model='SynGen', indexer=self.syn)
        self.vq = ExtAlgeb(src='vq', model='SynGen', indexer=self.syn)
```

BackRef for Connections

Groups enable backward references:

```
class Bus(Model):
    def __init__(self):
        # Collect all PQ loads connected to this bus
        self.PQ = BackRef()
```

(continues on next page)

(continued from previous page)

```
        # Collect all generators
        self.SynGen = BackRef()
```

Usage:

```
# Get list of PQ indices on bus 1
pq_indices = ss.Bus.PQ.v[0]  # List of PQ idx connected to first bus
```

Group Substitution

Replace models without changing input files:

```
# Original case uses GENCLS
# To use GENROU instead, load with model map
ss = andes.load('case.xlsx', model_map={'GENCLS': 'GENROU'})
```

Since both are in `SynGen` group, references work automatically.

Common Group Patterns

Generator Groups

```
StaticGen (base)
    ├── PV
    └── Slack

SynGen (dynamic generators)
    ├── GENCLS (classical)
    ├── GENROU (round-rotor)
    └── GENSAL (salient-pole)
```

Controller Groups

```
Exciter
    ├── ESST1A
    ├── ESDC2
    ├── SEXS
    └── ...

TurbineGov
    ├── TGOV1
    ├── IEEEG1
    ├── HYGOV
    └── ...
```

Group Requirements

When creating models for a group:

1. **Inherit group data class**: Use `ExciterData`, `TGOVData`, etc.
2. **Set group name**: `self.group = 'Exciter'`
3. **Provide required parameters**: Match `common_params`
4. **Match interface**: Variables and equations must be compatible

Creating New Groups

```
# In andes/models/group.py
class MyGroup(GroupBase):
    def __init__(self):
        super().__init__()
        self.common_params = ['bus', 'Sn']
        self.common_vars = ['P', 'Q']
```

Register in the models import file to make it available.

See Also

- *Parameters* - Parameter types including IdxParam
- *Variables* - External variables
- *Services* - BackRef service
- *Model Structure* - Complete model structure

4.4 Creating Models

Step-by-step guide to implementing new device models in ANDES.

4.4.1 Examples Overview

The examples progress from simple to advanced:

Example	Model	Complexity	Key Concepts
Example: Static Model (Shunt)	Shunt	Introductory	Static model, constant impedance behavior
Example: Dynamic Model (BusFreq)	BusFreq	Introductory	Dynamic model, transfer function blocks, signal processing
Example: TGOV1 Turbine Governor	TGOV1	Intermediate	Two implementation approaches (equation vs block), limiters
Example: IEEEST Power System Stabilizer	IEEEST	Advanced	External services, switchers, signal processing chain

4.4.2 Workflow

1. **Define ModelData class** - Declare parameters
2. **Define Model class** - Declare variables, services, equations
3. **Register with System** - Add to model registry
4. **Generate code** - Run `andes prepare`
5. **Test** - Validate against reference implementations

4.4.3 Model Structure

Every ANDES model follows a consistent structure with data definition, model logic, and registration.

See also:

Understanding these foundational concepts will help:

- *Atomic Types* - How parameters, variables, and services work as v-providers and e-providers
- *Hybrid Symbolic-Numeric Framework* - The symbolic-numeric framework

Basic Structure

A model consists of two classes: Data and Model.

```
from andes.core.model import Model, ModelData
from andes.core.param import NumParam, IdxParam
from andes.core.var import Algeb, State

class MyModelData(ModelData):
    """Data class defines parameters."""

    def __init__(self):
        super().__init__()
```

(continues on next page)

(continued from previous page)

```
        # Define parameters here
        self.bus = IdxParam(model='Bus', mandatory=True)
        self.Sn = NumParam(default=100, info='Rated power')

class MyModel(MyModelData, Model):
    """Model class defines variables and equations."""

    def __init__(self, system, config):
        MyModelData.__init__(self)
        Model.__init__(self, system, config)

        # Model flags
        self.flags.pflow = True
        self.flags.tds = True

        # Group assignment
        self.group = 'StaticLoad'

        # Services, variables, equations here
```

ModelData Class

Defines all input parameters:

```
class MyModelData(ModelData):
    def __init__(self):
        super().__init__()

        # Required parameters
        self.bus = IdxParam(model='Bus', mandatory=True)

        # Numerical parameters with defaults
        self.p0 = NumParam(default=0, info='Active power')
        self.q0 = NumParam(default=0, info='Reactive power')

        # Voltage and power base
        self.Vn = NumParam(default=110, info='Rated voltage')
        self.Sn = NumParam(default=100, info='Rated power')
```

Model Class

Defines services, variables, equations, and behaviors:

```
class MyModel(MyModelData, Model):
    def __init__(self, system, config):
        MyModelData.__init__(self)
        Model.__init__(self, system, config)
```

(continues on next page)

(continued from previous page)

```
        # 1. Flags - enable for specific analyses
        self.flags.pflow = True    # Power flow
        self.flags.tds = True      # Time-domain simulation
        self.flags.tds_init = True  # TDS initialization

        # 2. Group - for polymorphism
        self.group = 'StaticLoad'

        # 3. Services - intermediate calculations
        self.Zbase = ConstService(v_str='Vn**2/Sn')

        # 4. External parameters/variables
        self.v = ExtAlgeb(src='v', model='Bus', indexer=self.bus)

        # 5. Internal variables with equations
        self.P = Algeb(e_str='p0 - P')
```

Model Flags

Control which analyses include this model:

Flag	Analysis	Description
`pflow`	Power flow	Include in PFlow equations
`tds`	Time-domain	Include in TDS equations
`tds_init`	TDS init	Custom TDS initialization

```
self.flags.pflow = True
self.flags.tds = True
```

Component Order

Define components in this order for clarity:

1. **Flags and group**
2. **Config** (optional)
3. **ConstService** (constants)
4. **ExtService** (external constants)
5. **ExtParam** (external parameters)
6. **ExtVar** (external variables)
7. **VarService** (variable services)
8. **Discrete** (limiters, switches)

9. **Blocks** (transfer functions)
10. **Algeb** (algebraic variables)
11. **State** (state variables)

Connecting to Other Models

Reference via IdxParam

```
# In data class
self.bus = IdxParam(model='Bus', mandatory=True)
self.gen = IdxParam(model='StaticGen', mandatory=True)
```

Get External Parameters

```
self.Vn = ExtParam(src='Vn', model='Bus', indexer=self.bus)
```

Access External Variables

```
# Algebraic variable from Bus
self.v = ExtAlgeb(
    src='v',
    model='Bus',
    indexer=self.bus,
    e_str='-q'  # Inject reactive power
)
```

Equation Definition

Algebraic Equation

$0 = g(x, y)$

```
self.P = Algeb(
    v_str='p0',            # Initial value
    e_str='p0 - P',        # Equation: 0 = p0 - P
    info='Active power'
)
```

Differential Equation

$\dot{x} = f(x, y)$

```
self.omega = State(
    v_str='1.0',
    e_str='(Tm - Te) / M',  # d(omega)/dt = (Tm - Te) / M
    info='Rotor speed'
)
```

Complete Example

```
from andes.core.model import Model, ModelData
from andes.core.param import NumParam, IdxParam
from andes.core.var import Algeb, ExtAlgeb
from andes.core.service import ConstService

class PQData(ModelData):
    def __init__(self):
        super().__init__()
        self.bus = IdxParam(model='Bus', mandatory=True)
        self.Vn = NumParam(default=110, info='Rated voltage [kV]')
        self.p0 = NumParam(default=0, info='Active power [pu]')
        self.q0 = NumParam(default=0, info='Reactive power [pu]')

class PQ(PQData, Model):
    def __init__(self, system, config):
        PQData.__init__(self)
        Model.__init__(self, system, config)

        self.flags.pflow = True
        self.flags.tds = True
        self.group = 'StaticLoad'

        # External bus voltage angle and magnitude
        self.a = ExtAlgeb(
            src='a', model='Bus', indexer=self.bus,
            e_str='-p0 * u',
            info='Bus angle'
        )
        self.v = ExtAlgeb(
            src='v', model='Bus', indexer=self.bus,
            e_str='-q0 * u',
            info='Bus voltage'
        )
```

Registration

Add the model to `andes/models/__init__.py`:

```
from andes.models.static.pq import PQ

# In the models dictionary
file_classes = OrderedDict([
    # ...
    ('pq', ['PQ']),
    # ...
])
```

Code Generation

After creating or modifying a model:

```
andes prepare -i
```

This regenerates the numerical code for the new/modified model.

See Also

- *Example: Static Model (Shunt)* - Static model walkthrough (Shunt)
- *Example: Dynamic Model (BusFreq)* - Dynamic model walkthrough (BusFreq)
- *Example: TGOV1 Turbine Governor* - Advanced example with two implementation approaches
- *Example: IEEEST Power System Stabilizer* - Comprehensive example with signal switching
- *Testing Models* - Testing your models

4.4.4 Example: Static Model (Shunt)

This example walks through the real `Shunt` model from ANDES to explain static model concepts.

Background

Static models participate in power flow analysis and contribute algebraic equations to the DAE system. They do not have differential equations (state variables), making them simpler to implement than dynamic models. Common examples include loads, shunts, and static VAR compensators.

This tutorial examines the `Shunt` model—a phasor-domain shunt compensator. It demonstrates the constant impedance behavior where power varies with voltage squared ($P \propto V^2$, $Q \propto V^2$).

See also:

Before proceeding, ensure you understand:

- *Atomic Types* - How parameters and variables work as v-providers and e-providers
- *DAE Formulation* - Algebraic equations in the DAE system
- *Model Structure* - General model class structure

The Shunt Model

The complete model is located at `andes/models/shunt/shunt.py`. Here it is in full:

```
# File: andes/models/shunt/shunt.py

from andes.core import ModelData, IdxParam, NumParam, Model, ExtAlgeb


class ShuntData(ModelData):

    def __init__(self, system=None, name=None):
        super().__init__(system, name)

        self.bus = IdxParam(model='Bus', info="idx of connected bus",␣
↪mandatory=True)

        self.Sn = NumParam(default=100.0, info="Power rating", non_zero=True,␣
↪tex_name='S_n')
        self.Vn = NumParam(default=110.0, info="AC voltage rating", non_
↪zero=True, tex_name='V_n')
        self.g = NumParam(default=0.0, info="shunt conductance (real part)",␣
↪y=True, tex_name='g')
        self.b = NumParam(default=0.0, info="shunt susceptance (positive as␣
↪capacitive)", y=True, tex_name='b')
        self.fn = NumParam(default=60.0, info="rated frequency", tex_name='f_n
↪')


class ShuntModel(Model):
    """
    Shunt equations.
    """

    def __init__(self, system=None, config=None):
        Model.__init__(self, system, config)
        self.group = 'StaticShunt'
        self.flags.pflow = True
        self.flags.tds = True

        self.a = ExtAlgeb(model='Bus', src='a', indexer=self.bus, tex_name=r'\
↪theta',
                          ename='P',
                          tex_ename='P',
                          )
        self.v = ExtAlgeb(model='Bus', src='v', indexer=self.bus, tex_name='V
```

(continues on next page)

(continued from previous page)

```
↪',
                            ename='Q',
                            tex_ename='Q',
                            )

        self.a.e_str = 'u * v**2 * g'
        self.v.e_str = '-u * v**2 * b'


class Shunt(ShuntData, ShuntModel):
    """
    Phasor-domain shunt compensator Model.
    """

    def __init__(self, system=None, config=None):
        ShuntData.__init__(self)
        ShuntModel.__init__(self, system, config)
```

Step-by-Step Breakdown

Step 1: Data Class

The `ShuntData` class defines all input parameters:

```
class ShuntData(ModelData):

    def __init__(self, system=None, name=None):
        super().__init__(system, name)

        # Connection to the network
        self.bus = IdxParam(model='Bus', info="idx of connected bus",␣
↪mandatory=True)

        # Ratings
        self.Sn = NumParam(default=100.0, info="Power rating", non_zero=True,␣
↪tex_name='S_n')
        self.Vn = NumParam(default=110.0, info="AC voltage rating", non_
↪zero=True, tex_name='V_n')

        # Shunt admittance components
        self.g = NumParam(default=0.0, info="shunt conductance (real part)",␣
↪y=True, tex_name='g')
        self.b = NumParam(default=0.0, info="shunt susceptance (positive as␣
↪capacitive)", y=True, tex_name='b')

        self.fn = NumParam(default=60.0, info="rated frequency", tex_name='f_n
↪')
```

Key observations:

- **IdxParam**: References another model (`Bus`) by index
- **NumParam**: Numerical parameters with defaults, units, and TeX names
- **y=True**: Marks parameters as part of the admittance matrix (for sparse solvers)
- **mandatory=True**: Parameter must be provided in input data

Step 2: Model Class

The `ShuntModel` class defines the behavior:

```
class ShuntModel(Model):

    def __init__(self, system=None, config=None):
        Model.__init__(self, system, config)

        # Group assignment for polymorphism
        self.group = 'StaticShunt'

        # Enable for power flow and time-domain simulation
        self.flags.pflow = True
        self.flags.tds = True
```

- **group**: Assigns to `StaticShunt` group for classification
- **flags.pflow**: Include in power flow equations
- **flags.tds**: Include in time-domain simulation

Step 3: External Variables

Connect to bus voltage variables:

```
        self.a = ExtAlgeb(model='Bus', src='a', indexer=self.bus, tex_name=r'\
↪theta',
                          ename='P',
                          tex_ename='P',
                          )
        self.v = ExtAlgeb(model='Bus', src='v', indexer=self.bus, tex_name='V
↪',
                          ename='Q',
                          tex_ename='Q',
                          )
```

- **ExtAlgeb**: External algebraic variable from another model
- **model='Bus'**: The source model
- **src='a'** / **src='v'**: The source variable name (angle / voltage magnitude)
- **indexer=self.bus**: Which bus instances to link to

- **ename** / **tex_ename**: Equation name for output

Step 4: Power Injection Equations

The shunt admittance equations:

```
        self.a.e_str = 'u * v**2 * g'
        self.v.e_str = '-u * v**2 * b'
```

These implement the constant impedance power equations:

$$P = V^2 \cdot g$$

$$Q = -V^2 \cdot b$$

Where:

- g is conductance (positive = absorbs active power)
- b is susceptance (positive = capacitive, generates reactive power)
- The negative sign on Q follows the generator convention (capacitor injects Q)
- **u** is the online status (inherited from `Model`)

Step 5: Combined Class

The final class uses multiple inheritance:

```
class Shunt(ShuntData, ShuntModel):
    """
    Phasor-domain shunt compensator Model.
    """

    def __init__(self, system=None, config=None):
        ShuntData.__init__(self)
        ShuntModel.__init__(self, system, config)
```

This pattern separates:

- **Data definition** (parameters) in the `*Data` class
- **Model behavior** (variables, equations) in the `*Model` class

Key Concepts Illustrated

1. Parameters as v-providers

All parameters have a `v` attribute containing their values:

```
# At runtime, for 3 shunt devices:
# self.g.v = np.array([0.01, 0.02, 0.015])
# self.b.v = np.array([0.05, 0.10, 0.08])
```

2. External variables as e-providers

`ExtAlgeb` links to variables in other models and contributes equations:

```
self.a.e_str = 'u * v**2 * g'
```

When ANDES evaluates equations:

1. Substitutes `u`, `v`, `g` with their `.v` arrays (all are v-providers)
2. Stores result in `self.a.e` (equation contribution)
3. Accumulates into global DAE array at addresses `self.a.a`

3. Constant Impedance Behavior

The V^2 dependence is the defining characteristic:

- When voltage drops, power consumption drops quadratically
- This is more stable than constant power loads during voltage disturbances
- Critical for voltage stability studies

4. Sign Convention

- **Positive injection** = power flowing into the bus (generation)
- **Negative injection** = power flowing out of the bus (load)
- Capacitive shunt ($b > 0$): $Q = -V^2 b < 0$ means reactive power is injected (generated)

Testing the Model

```
import andes

ss = andes.load('ieee14.xlsx')

# Check existing shunts
print(ss.Shunt.as_df())

# Run power flow
ss.PFlow.run()

# Check reactive power injection
print(f"Shunt Q injection: {ss.Shunt.v.e}")
```

See Also

- *Model Structure* - General model structure
- *Example: Dynamic Model (BusFreq)* - Dynamic model with states
- *Parameters* - Parameter types
- *Variables* - Variable types and the v-e-a triad
- Source code: `andes/models/shunt/shunt.py`

4.4.5 Example: Dynamic Model (BusFreq)

This example walks through the real `BusFreq` model from ANDES to explain dynamic model concepts.

Background

Dynamic models participate in time-domain simulation (TDS) and contribute both differential and algebraic equations to the DAE system. They have state variables whose time derivatives define system dynamics.

This tutorial examines the `BusFreq` model—a bus frequency measurement device. It demonstrates key concepts of dynamic modeling: transfer function blocks, external variable coupling, and signal processing. Unlike controllers that write back to other devices, this model reads bus voltage angle and computes local frequency.

See also:

Before proceeding, ensure you understand:

- *Example: Static Model (Shunt)* - Static model basics (start here if new)
- *Atomic Types* - v-providers and e-providers
- *DAE Formulation* - Differential equations in the DAE system
- *Blocks* - Transfer function blocks

The BusFreq Model

The complete model is located at `andes/models/measurement/busfreq.py`. Here it is in full:

```
# File: andes/models/measurement/busfreq.py

"""
Bus frequency estimation based on high-pass filter.
"""

from andes.core import (ModelData, Model, IdxParam, NumParam,
                        ConstService, ExtService, Algeb, ExtAlgeb,
                        Lag, Washout)


class BusFreq(ModelData, Model):
    """
    Bus frequency measurement. Outputs frequency in per unit value.

    The bus frequency output variable is `f`.
    The frequency deviation variable is `WO_y`.
    """

    def __init__(self, system, config):
        ModelData.__init__(self)
        Model.__init__(self, system, config)
        self.flags.tds = True
        self.group = 'FreqMeasurement'

        # Parameters
        self.bus = IdxParam(info="bus idx", mandatory=True)

        self.Tf = NumParam(default=0.02,
                           info="input digital filter time const",
                           unit="sec",
                           tex_name='T_f',
                           )
        self.Tw = NumParam(default=0.1,
                           info="washout time const",
                           unit="sec",
                           tex_name='T_w',
                           )
        self.fn = NumParam(default=60.0,
                           info="nominal frequency",
                           unit='Hz',
                           tex_name='f_n',
                           )

        # Variables
        self.iwn = ConstService(v_str='u / (2 * pi * fn)', tex_name=r'1/\
↪omega_n')
        self.a0 = ExtService(src='a',
                             model='Bus',
```

(continues on next page)

(continued from previous page)

```
                                indexer=self.bus,
                                tex_name=r'\theta_0',
                                info='initial phase angle',
                                )
        self.a = ExtAlgeb(model='Bus',
                          src='a',
                          indexer=self.bus,
                          tex_name=r'\theta',
                          )
        self.v = ExtAlgeb(model='Bus',
                          src='v',
                          indexer=self.bus,
                          tex_name=r'V',
                          )
        self.L = Lag(u='(a-a0)',
                     T=self.Tf,
                     K=1,
                     info='digital filter',
                     )
        # the output `WO_y` is the frequency deviation in p.u.
        self.WO = Washout(u=self.L_y,
                          K=self.iwn,
                          T=self.Tw,
                          info='angle washout',
                          )
        self.WO_y.info = 'frequency deviation'
        self.WO_y.unit = 'p.u. (Hz)'

        self.f = Algeb(info='frequency output',
                       unit='p.u. (Hz)',
                       tex_name='f',
                       v_str='1',
                       e_str='1 + WO_y - f',
                       )
```

Step-by-Step Breakdown

Step 1: Class Structure

Unlike the Shunt model, BusFreq combines data and model in a single class:

```
class BusFreq(ModelData, Model):
    def __init__(self, system, config):
        ModelData.__init__(self)
        Model.__init__(self, system, config)
        self.flags.tds = True          # Enable for time-domain simulation
        self.group = 'FreqMeasurement'
```

Key observations:

- **`flags.tds = True`**: Marks this as a dynamic model for TDS
- **`group`**: Assigns to `FreqMeasurement` group
- No `flags.pflow` because frequency measurement is not meaningful in steady-state

Step 2: Parameters

```
        # Connection
        self.bus = IdxParam(info="bus idx", mandatory=True)

        # Time constants
        self.Tf = NumParam(default=0.02, info="input digital filter time const
↪", unit="sec")
        self.Tw = NumParam(default=0.1, info="washout time const", unit="sec")

        # Nominal frequency
        self.fn = NumParam(default=60.0, info="nominal frequency", unit='Hz')
```

- **`bus`**: References which bus to measure
- **`Tf`**: Low-pass filter time constant for noise rejection
- **`Tw`**: Washout filter time constant for frequency extraction
- **`fn`**: Nominal system frequency (60 Hz in North America)

Step 3: Services

```
        self.iwn = ConstService(v_str='u / (2 * pi * fn)', tex_name=r'1/\
↪omega_n')
        self.a0 = ExtService(src='a', model='Bus', indexer=self.bus,
                             tex_name=r'\theta_0', info='initial phase angle')
```

- **`ConstService`**: Calculates inverse of angular frequency: $\frac{1}{\omega_n} = \frac{1}{2\pi f_n}$
- **`ExtService`**: Captures the initial bus angle from power flow solution

The `u` in `v_str='u / (2 * pi * fn)'` is the online status flag (inherited from Model). When `u=0` (offline), the service evaluates to zero.

Step 4: External Variables

```
        self.a = ExtAlgeb(model='Bus', src='a', indexer=self.bus, tex_name=r'\
↪theta')
        self.v = ExtAlgeb(model='Bus', src='v', indexer=self.bus, tex_name=r'V
↪')
```

- **`ExtAlgeb`**: Links to algebraic variables from the Bus model

- **a**: Bus voltage angle (radians)
- **v**: Bus voltage magnitude (per unit)

Unlike the Shunt model, BusFreq does not inject power—it only reads the bus angle.

Step 5: Transfer Function Blocks

This is where dynamic behavior is defined:

```
        self.L = Lag(u='(a-a0)', T=self.Tf, K=1, info='digital filter')
        self.WO = Washout(u=self.L_y, K=self.iwn, T=self.Tw, info='angle␣
↪washout')
```

Lag Block

The `Lag` block implements a first-order low-pass filter:

$$\frac{y}{u} = \frac{K}{1 + sT}$$

- **Input**: `(a - a0)` — angle deviation from initial value
- **Output**: `L_y` — filtered angle deviation
- **Purpose**: Removes high-frequency noise from angle measurement

Washout Block

The `Washout` block extracts rate of change (high-pass filter):

$$\frac{y}{u} = \frac{sKT}{1 + sT}$$

- **Input**: `L_y` — filtered angle deviation
- **Output**: `WO_y` — frequency deviation in per unit
- **Purpose**: Converts angle to frequency ($\omega = \frac{d\theta}{dt}$)

The blocks create internal state variables automatically. ANDES uses naming conventions:

- `L_y` is the output of block `L`
- `WO_y` is the output of block `WO`

Step 6: Output Variable

```
        self.f = Algeb(info='frequency output', unit='p.u. (Hz)', tex_name='f
↪',
                       v_str='1',
                       e_str='1 + WO_y - f')
```

The final frequency output:

- **`v_str='1'`**: Initialize to 1.0 per unit (nominal frequency)
- **`e_str='1 + WO_y - f'`**: Equation $0 = 1 + \Delta\omega - f$, giving $f = 1 + \Delta\omega$

Signal Flow

The signal processing chain:

```
Bus angle (a) → Subtract initial (a - a0) → Lag filter → Washout → Add 1.0 → f
                                                ↓              ↓
                                              L_y            WO_y
```

1. Read bus angle `a` from the Bus model
2. Subtract initial angle `a0` to get angle deviation
3. Apply low-pass filter (`Lag`) for noise rejection
4. Apply washout to get frequency deviation (derivative of angle)
5. Add nominal frequency (1.0 p.u.) to get absolute frequency

Key Concepts Illustrated

1. Transfer Function Blocks

Blocks encapsulate state variables and equations:

```
self.L = Lag(u='(a-a0)', T=self.Tf, K=1)
```

Internally, this creates:

- A state variable for the filter state
- Differential equation: $\dot{x} = \frac{K \cdot u - x}{T}$
- Output: `L_y = x`

2. Block Chaining

Blocks can be chained by referencing outputs:

```
self.L = Lag(u='(a-a0)', ...)
self.WO = Washout(u=self.L_y, ...)  # L_y is output of L
```

3. Dynamic vs Static Flags

```
self.flags.tds = True    # Include in time-domain simulation
# No flags.pflow         # Not included in power flow
```

4. External Services for Initialization

```
self.a0 = ExtService(src='a', model='Bus', indexer=self.bus)
```

Captures the power flow solution value for use during TDS. Without this, the model wouldn't know the initial bus angle.

Testing the Model

```
import andes

ss = andes.load('kundur_full.xlsx', setup=False)

# Add BusFreq to bus 1
ss.add('BusFreq', bus=1, Tf=0.02, Tw=0.1)

ss.setup()
ss.PFlow.run()
ss.TDS.config.tf = 10
ss.TDS.run()

# Plot frequency measurement
ss.TDS.plt.plot(ss.BusFreq.f)
```

Key Differences from Static Model

Aspect	Static Model (Shunt)	Dynamic Model (BusFreq)
Flags	`pflow=True, tds=True`	`tds=True` only
Behavior	Algebraic equations	Transfer function blocks
State	None	Internal block states
Purpose	Power injection	Signal processing
External vars	Write to bus (e_str)	Read only

See Also

- *Model Structure* - General structure
- *Example: Static Model (Shunt)* - Static model example (Shunt)
- *Example: TGOV1 Turbine Governor* - Governor model with explicit state variables
- *Example: IEEEST Power System Stabilizer* - Advanced controller with signal switching
- *Blocks* - Transfer function blocks reference
- Source code: `andes/models/measurement/busfreq.py`

4.4.6 Example: TGOV1 Turbine Governor

This example demonstrates modeling a turbine governor using both equation-based and block-based approaches. TGOV1 is a commonly used turbine governor model that is sufficiently complex to illustrate key modeling concepts.

Model Overview

The TGOV1 turbine governor model consists of:

- A droop characteristic
- A first-order lag with anti-windup limiter
- A lead-lag transfer function

Block Diagram

```
                ┌──────────────────────────────────────────────────────────┐
                │                                                          │
ωref ─+─────────┼──►(×)──►┌─────────┐   ┌──────────────┐   ┌────────────┐   │
      │         │   1/R   │ Lag     │──►│ Lead-Lag     │──►│   + Dt     │───┼──► Pout
ω ────┴─────────┘         │ K/T1    │   │ (T2,T3)      │   └────────────┘   │
                          │ VMIN    │   └──────────────┘          ▲         │
                          │ VMAX    │                             │         │
                          └─────────┘                             │         │
                          └──────────────────────────────────────┘         │
                                                                           ▼
                                                                          tm
```

Mathematical Equations

The model is described by two differential equations and six algebraic equations:

Differential Equations:

$$\begin{bmatrix} \dot{x}_{LG} \\ \dot{x}_{LL} \end{bmatrix} = \begin{bmatrix} z_{i,lim}^{LG}(P_d - x_{LG})/T_1 \\ (x_{LG} - x_{LL})/T_3 \end{bmatrix}$$

Algebraic Equations:

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} (1-\omega) - \omega_d \\ R \times \tau_{m0} - P_{ref} \\ (P_{ref} + \omega_d)/R - P_d \\ D_t\omega_d + y_{LL} - P_{OUT} \\ \frac{T_2}{T_3}(x_{LG} - x_{LL}) + x_{LL} - y_{LL} \\ u(P_{OUT} - \tau_{m0}) \end{bmatrix}$$

Where:

- x_{LG}, x_{LL} are internal states of the lag and lead-lag blocks
- y_{LL} is the lead-lag output
- ω is generator speed, ω_d is generator under-speed
- P_d is droop output, τ_{m0} is steady-state torque input
- P_{OUT} is turbine output summed at the generator

Equation-Based Implementation

The following code implements TGOV1 using explicit equations. This approach gives complete control over the mathematical formulation.

```
def __init__(self, system, config):
    # 1. Declare parameters from case file inputs
    self.R = NumParam(info='Turbine governor droop',
                      non_zero=True, ipower=True)
    self.T1 = NumParam(info='Lag time constant')
    self.T2 = NumParam(info='Lead time constant')
    self.T3 = NumParam(info='Lag time constant')
    self.Dt = NumParam(info='Damping coefficient')
    self.VMIN = NumParam(info='Minimum gate limit')
    self.VMAX = NumParam(info='Maximum gate limit')

    # 2. Declare external variables from generators
    self.omega = ExtState(src='omega',
                          model='SynGen',
                          indexer=self.syn,
                          info='Generator speed')
    self.tm = ExtAlgeb(src='tm',
                       model='SynGen',
```

(continues on next page)

(continued from previous page)

```
                    indexer=self.syn,
                    e_str='u*(pout-tm0)',
                    info='Generator torque input')

    # 3. Declare initial values from generators
    self.tm0 = ExtService(src='tm',
                          model='SynGen',
                          indexer=self.syn,
                          info='Initial torque input')

    # 4. Declare variables and equations
    self.pref = Algeb(info='Reference power input',
                      v_str='tm0*R',
                      e_str='tm0*R-pref')

    self.wd = Algeb(info='Generator under speed',
                    e_str='(1-omega)-wd')

    self.pd = Algeb(info='Droop output',
                    v_str='tm0',
                    e_str='(wd+pref)/R-pd')

    self.LG_x = State(info='State in the lag TF',
                      v_str='pd',
                      e_str='LG_lim_zi*(pd-LG_x)/T1')

    self.LG_lim = AntiWindup(u=self.LG_x,
                             lower=self.VMIN,
                             upper=self.VMAX)

    self.LL_x = State(info='State in the lead-lag TF',
                      v_str='LG_x',
                      e_str='(LG_x-LL_x)/T3')

    self.LL_y = Algeb(info='Lead-lag Output',
                      v_str='LG_x',
                      e_str='T2/T3*(LG_x-LL_x)+LL_x-LL_y')

    self.pout = Algeb(info='Turbine output power',
                      v_str='tm0',
                      e_str='(LL_y+Dt*wd)-pout')
```

Key observations:

- `ExtState` and `ExtAlgeb` link to generator variables
- `ExtService` retrieves initial values for initialization
- `AntiWindup` discrete component provides limiter flags
- The `LG_lim_zi` flag in `e_str` implements anti-windup behavior

Block-Based Implementation

The same model can be implemented more concisely using transfer function blocks. This approach improves readability and reduces the chance of equation errors.

```
def __init__(self, system, config):
    TGBase.__init__(self, system, config)

    # Calculate gain from droop
    self.gain = ConstService(v_str='u/R')

    # Reference power
    self.pref = Algeb(info='Reference power input',
                      tex_name='P_{ref}',
                      v_str='tm0 * R',
                      e_str='tm0 * R - pref')

    # Generator under-speed
    self.wd = Algeb(info='Generator under speed',
                    unit='p.u.',
                    tex_name=r'\omega_{dev}',
                    v_str='0',
                    e_str='(wref - omega) - wd')

    # Droop output
    self.pd = Algeb(info='Pref plus under speed times gain',
                    unit='p.u.',
                    tex_name="P_d",
                    v_str='u * tm0',
                    e_str='u*(wd + pref + paux) * gain - pd')

    # Lag with anti-windup (replaces manual LG_x and LG_lim)
    self.LAG = LagAntiWindup(u=self.pd,
                             K=1,
                             T=self.T1,
                             lower=self.VMIN,
                             upper=self.VMAX)

    # Lead-lag block (replaces manual LL_x and LL_y)
    self.LL = LeadLag(u=self.LAG_y,
                      T1=self.T2,
                      T2=self.T3)

    # Output equation
    self.pout.e_str = '(LL_y + Dt * wd) - pout'
```

Key observations:

- `LagAntiWindup` encapsulates the lag transfer function with anti-windup
- `LeadLag` encapsulates the lead-lag transfer function
- Block outputs follow the pattern `BlockName_y`

- The block-based code is shorter and easier to validate against the diagram

Comparison of Approaches

Aspect	Equation-Based	Block-Based
Lines of code	More	Fewer
Readability	Requires understanding equations	Matches block diagram
Flexibility	Full control	Limited to block capabilities
Error risk	Higher (manual equations)	Lower (validated blocks)
Performance	Same	Same

Guidelines

- **Use blocks** when the model follows a standard control block diagram
- **Use equations** when implementing non-standard transfer functions or when blocks don't exist
- **Combine both** approaches when needed - blocks can be mixed with custom equations

Complete Source Code

The complete implementations can be found in:

- Equation-based: `andes/models/governor/tgov1.py` (class `TGOV1ModelAlt`)
- Block-based: `andes/models/governor/tgov1.py` (class `TGOV1Model`)

See Also

- *Blocks* - Available transfer function blocks
- *Discrete Components* - Discrete components including AntiWindup
- *Example: IEEEST Power System Stabilizer* - More complex example with input selection

4.4.7 Example: IEEEST Power System Stabilizer

This comprehensive example walks through the complete implementation of the IEEEST power system stabilizer model. It demonstrates advanced techniques including optional parameters, input selection, external services, and mode switching.

Model Overview

IEEEST is a versatile PSS model with:

- Multiple input signal options (speed, frequency, power, voltage)
- Two stages of lead-lag compensation
- Gain and washout blocks
- Output limiting

Block Diagram

```
                                                                LSMAX
                                                                  |
Signal ──►[F1: 2nd-order]──►[F2: Lead-Lag 2nd]──►                 ↓
  |            Lag                 order
↪[LL1]──►[LL2]──►[×Ks]──►[WO]──►[Lim]──► Vss
  |
↪    |
  |
↪    ↓
  ▼
↪ LSMIN
MODE selects:
1: ω-1 (speed deviation)
2: f-1 (frequency deviation)
3: Pe/SnSb (electrical power)
4: Pm-Pm0 (accelerating power)
5: V (bus voltage)
6: dV/dt (voltage derivative)
```

Data Classes

PSSBaseData

`PSSBaseData` holds parameters shared by all PSS models:

```
class PSSBaseData(ModelData):
    def __init__(self):
        super().__init__()
        # Link to exciter (required for all PSS)
        self.avr = IdxParam(model='Exciter',
                            mandatory=True,
                            info='Exciter idx')
```

IEEESTData

`IEEESTData` defines IEEEST-specific parameters:

```
class IEEESTData(PSSBaseData):
    def __init__(self):
        PSSBaseData.__init__(self)

        # Input mode selection
        self.MODE = NumParam(default=1,
                             info='Input signal mode (1-6)')

        # Optional remote bus
        self.busr = IdxParam(model='Bus',
                             info='Remote bus idx',
                             default=None)

        # Transfer function parameters
        self.A1 = NumParam(default=0, info='2nd-order filter coeff.')
        self.A2 = NumParam(default=0, info='2nd-order filter coeff.')
        # ... additional parameters
```

PSSBase Model Class

`PSSBase` defines common external connections shared by all PSS models:

```
class PSSBase(Model):
    def __init__(self, system, config):
        super().__init__(system, config)

        # Register with PSS group and enable TDS
        self.group = 'PSS'
        self.flags.tds = True
```

Important: Setting `self.flags.tds = True` is required to include the model in time-domain simulation. Without this flag, data loads but variables receive no addresses and equations are skipped.

Parameter Replacement

The `Replace` service replaces input values that satisfy a condition:

```
# Replace zero limits with large values (effectively no limit)
self.VCUr = Replace(self.VCU, lambda x: np.equal(x, 0.0), 999)
self.VCLr = Replace(self.VCL, lambda x: np.equal(x, 0.0), -999)
```

Retrieving Connected Devices

PSS needs to find the generator connected through the exciter:

```
# Get generator idx from exciter
self.syn = ExtParam(model='Exciter',
                    src='syn',
                    indexer=self.avr,
                    export=False,
                    info='Retrieved generator idx',
                    vtype=str)

# Get bus from generator
self.bus = ExtParam(model='SynGen',
                    src='bus',
                    indexer=self.syn,
                    export=False,
                    info='Retrieved bus idx',
                    vtype=str,
                    default=None)
```

Optional Remote Bus

PSS models support an optional remote bus. `DataSelect` chooses between the optional input and a fallback:

```
# Use busr if provided, otherwise use local bus
self.buss = DataSelect(self.busr,
                       self.bus,
                       info='selected bus (bus or busr)')
```

Device Finding

`DeviceFinder` locates or creates measurement devices:

```
# Find or create frequency measurement device for the selected bus
self.busfreq = DeviceFinder(self.busf,
                            link=self.buss,
                            idx_name='bus')
```

If `busf` is not specified in the input data, `DeviceFinder` will find an existing `BusFreq` device for the bus or create one.

IEEESTModel

Configuration

Add model-specific configuration options:

```
self.config.add(OrderedDict([('freq_model', 'BusFreq')]))
self.config.add_extra('_help', {'freq_model': 'default freq. measurement model
↪'})
self.config.add_extra('_alt', {'freq_model': ('BusFreq',)})

# Set the measurement model for DeviceFinder
self.busf.model = self.config.freq_model
```

Voltage Derivative

For input mode 6, we need dV/dt. `Derivative` computes finite differences:

```
self.dv = Derivative(self.v,
                     tex_name='dV/dt',
                     info='Finite difference of bus voltage')
```

Per-Unit Conversion

Retrieve the machine-to-system base conversion factor:

```
self.SnSb = ExtService(model='SynGen',
                       src='M',
                       indexer=self.syn,
                       attr='pu_coeff',
                       info='Machine base to sys base factor for power',
                       tex_name='(Sb/Sn)')
```

Note the `attr='pu_coeff'` - this accesses a special attribute of the `M` variable that stores the per-unit conversion coefficient.

Input Mode Selection with Switcher

`Switcher` parses the input mode into boolean flags:

```
self.SW = Switcher(u=self.MODE,
                   options=[0, 1, 2, 3, 4, 5, 6])
```

This creates flags `SW_s0` through `SW_s6`. We include `0` for padding so that `SW_s1` corresponds to MODE 1.

Input Signal Selection

The input signal uses piece-wise construction based on mode:

```
self.sig = Algeb(tex_name='S_{ig}',
                 info='Input signal')

# Initial value (uses MODE-dependent expression)
self.sig.v_str = ('SW_s1*(omega-1) + SW_s2*0 + SW_s3*(tm0/SnSb) + '
                  'SW_s4*(tm-tm0) + SW_s5*v + SW_s6*0')

# Equation (uses MODE-dependent expression)
self.sig.e_str = ('SW_s1*(omega-1) + SW_s2*(f-1) + SW_s3*(te/SnSb) + '
                  'SW_s4*(tm-tm0) + SW_s5*v + SW_s6*dv_v - sig')
```

Key observations:

- `v_str` and `e_str` are assigned separately for readability
- Each mode is multiplied by its corresponding flag (`SW_s1`, `SW_s2`, etc.)
- Only the active mode contributes to the sum

Transfer Function Blocks

The signal processing chain uses nested blocks:

```
# Second-order lag filter
self.F1 = Lag2ndOrd(u=self.sig, K=1, T1=self.A1, T2=self.A2)

# Second-order lead-lag
self.F2 = LeadLag2ndOrd(u=self.F1_y,
                        T1=self.A3, T2=self.A4,
                        T3=self.A5, T4=self.A6,
                        zero_out=True)

# First lead-lag stage
self.LL1 = LeadLag(u=self.F2_y, T1=self.T1, T2=self.T2, zero_out=True)

# Second lead-lag stage
self.LL2 = LeadLag(u=self.LL1_y, T1=self.T3, T2=self.T4, zero_out=True)

# Gain
self.Vks = Gain(u=self.LL2_y, K=self.KS)

# Washout or lag (depending on T5/T6)
self.WO = WashoutOrLag(u=self.Vks_y,
                       T=self.T6,
                       K=self.T5,
                       name='WO',
                       zero_out=True)
```

Output Limiting

Two-stage limiting with algebraic variable:

```
# Internal signal limiter
self.VLIM = Limiter(u=self.WO_y,
                    lower=self.LSMIN,
                    upper=self.LSMAX,
                    info='Vss limiter')

# Vss applies the limiter
self.Vss = Algeb(tex_name='V_{ss}',
                 info='Voltage output before output limiter',
                 e_str='VLIM_zi * WO_y + VLIM_zu * LSMAX + VLIM_zl * LSMIN -␣
↪Vss')

# Output limiter based on bus voltage
self.OLIM = Limiter(u=self.v,
                    lower=self.VCLr,
                    upper=self.VCUr,
                    info='output limiter')

# Final output (defined in PSSBase, equation assigned here)
self.vsout.e_str = 'OLIM_zi * Vss - vsout'
```

Final Assembly

Combine data and model classes:

```
class IEEEST(IEEESTData, IEEESTModel):
    def __init__(self, system, config):
        IEEESTData.__init__(self)
        IEEESTModel.__init__(self, system, config)
```

Model Registration

Adding to Model List

Edit `andes/models/__init__.py`:

```
file_classes = list([
    ...
    ('pss', ['IEEEST', 'ST2CUT']),
    ...
])
```

Group Definition

Ensure the PSS group exists in `andes/models/group.py`:

```
class PSS(GroupBase):
    """Power system stabilizer group."""

    def __init__(self):
        super().__init__()
        self.common_vars.extend(('vsout',))
```

All PSS models must have `vsout` as a common variable.

Key Techniques Summary

Technique	Component	Purpose
`Replace`	Service	Substitute invalid input values
`ExtParam`	Parameter	Retrieve parameters from linked models
`DataSelect`	Service	Choose between optional and fallback values
`DeviceFinder`	Service	Find or create linked devices
`Derivative`	Discrete	Compute finite differences
`Switcher`	Discrete	Parse mode into boolean flags
`zero_out`	Block option	Output zero when time constant is zero

Complete Source Code

The complete implementation is in `andes/models/pss/ieeest.py`.

See Also

- *Example: TGOV1 Turbine Governor* - Simpler example with two implementation styles
- *Services* - Services including DataSelect and DeviceFinder
- *Discrete Components* - Discrete components including Switcher
- *Blocks* - Transfer function blocks

4.4.8 Testing Models

After creating a new model, thorough testing ensures correctness and robustness.

Testing Levels

1. **Unit tests**: Model instantiation and basic behavior
2. **Integration tests**: Model works within full system
3. **Verification tests**: Results match reference implementations

Quick Validation

Check Model Loads

```
import andes

ss = andes.System()

# Check model is registered
assert 'MyModel' in ss.models

# Check model documentation
print(ss.MyModel.doc())
```

Check Parameters

```
ss = andes.load('case.xlsx', setup=False)

# Add test device
ss.add('MyModel', bus=1, param1=0.5)

ss.setup()

# Verify parameters loaded
print(ss.MyModel.as_df())
```

Power Flow Test

For models with `flags.pflow = True`:

```
import andes

ss = andes.load('ieee14.xlsx')

# Run power flow
ss.PFlow.run()

# Check convergence
assert ss.PFlow.converged

# Check results are reasonable
v = ss.dae.y[ss.Bus.v.a]
assert all(v > 0.9) and all(v < 1.1)
```

Time-Domain Test

For models with `flags.tds = True`:

Flat Run

System should stay at equilibrium without disturbances:

```
ss = andes.load('case.xlsx')
ss.PFlow.run()

# Run without disturbances
ss.TDS.config.tf = 2
ss.TDS.run()

# Check states stayed constant using get_timeseries()
omega_df = ss.TDS.get_timeseries(ss.GENROU.omega)
deviation = omega_df.max().max() - omega_df.min().min()

assert deviation < 1e-6, "States should not change in flat run"
```

Disturbance Response

```
ss = andes.load('case.xlsx', setup=False)

# Add disturbance
ss.add('Fault', bus=3, tf=1.0, tc=1.1)

ss.setup()
```

(continues on next page)

(continued from previous page)

```
ss.PFlow.run()
ss.TDS.config.tf = 5
ss.TDS.run()

# Check simulation completed
assert ss.TDS.exit_code == 0

# Check reasonable response
omega_df = ss.TDS.get_timeseries(ss.GENROU.omega)
assert omega_df.max().max() < 1.1, "Speed deviation too large"
```

Initialization Test

Check that initial conditions satisfy equations:

```
ss = andes.load('case.xlsx')
ss.PFlow.run()
ss.TDS.init()

# Check equation residuals are small
f_max = abs(ss.dae.f).max()
g_max = abs(ss.dae.g).max()

assert f_max < 1e-6, f"Differential equations not satisfied: {f_max}"
assert g_max < 1e-6, f"Algebraic equations not satisfied: {g_max}"
```

Writing pytest Tests

Create test file `tests/test_mymodel.py`:

```
import pytest
import andes

class TestMyModel:
    """Tests for MyModel."""

    def test_instantiation(self):
        """Model can be instantiated."""
        ss = andes.System()
        assert 'MyModel' in ss.models

    def test_pflow(self):
        """Power flow converges with MyModel."""
        ss = andes.load('test_case.xlsx')
        ss.PFlow.run()
        assert ss.PFlow.converged

    def test_tds_flat(self):
```

(continues on next page)

(continued from previous page)

```
        """TDS flat run is stable."""
        ss = andes.load('test_case.xlsx')
        ss.PFlow.run()
        ss.TDS.config.tf = 2
        ss.TDS.run()

        omega_df = ss.TDS.get_timeseries(ss.GENROU.omega)
        assert omega_df.std().max() < 1e-6

    def test_tds_fault(self):
        """TDS with fault completes."""
        ss = andes.load('test_case.xlsx', setup=False)
        ss.add('Fault', bus=3, tf=1.0, tc=1.1)
        ss.setup()
        ss.PFlow.run()
        ss.TDS.config.tf = 5
        ss.TDS.run()

        assert ss.TDS.exit_code == 0
```

Run tests:

```
pytest tests/test_mymodel.py -v
```

Verification Against Reference

Compare with PSS/E

1. Run same case in PSS/E
2. Apply identical disturbance
3. Export time series
4. Compare key variables

```
import numpy as np
import pandas as pd

# Load ANDES results
omega_df = ss.TDS.get_timeseries(ss.GENROU.omega)
andes_omega = omega_df.iloc[:, 0].values  # First generator
andes_t = omega_df.index.values

# Load PSS/E results
psse = pd.read_csv('psse_results.csv')

# Interpolate to common time base
from scipy.interpolate import interp1d
f = interp1d(psse['time'], psse['omega'], fill_value='extrapolate')
```

(continues on next page)

(continued from previous page)

```
psse_omega = f(andes_t)

# Compare
error = np.abs(andes_omega - psse_omega).max()
assert error < 0.01, f"Max error: {error}"
```

Visual Comparison

```
import matplotlib.pyplot as plt

fig, ax = plt.subplots()
ax.plot(andes_t, andes_omega, label='ANDES')
ax.plot(psse['time'], psse['omega'], '--', label='PSS/E')
ax.set_xlabel('Time [s]')
ax.set_ylabel('Speed [pu]')
ax.legend()
plt.savefig('verification.png')
```

Debugging Tips

Check Variable Values

```
# After init
print(f"omega initial: {ss.GENROU.omega.v}")
print(f"Pv initial: {ss.TGOVSimple.Pv.v}")
```

Check Equation Residuals

```
# After equation update
ss.f_update()
ss.g_update()

print(f"Max f residual: {abs(ss.dae.f).max()}")
print(f"Max g residual: {abs(ss.dae.g).max()}")
```

Trace Jacobian Issues

```
# Check Jacobian structure
ss.j_update()

import matplotlib.pyplot as plt
plt.spy(ss.dae.gy)
plt.title('Jacobian gy sparsity')
plt.savefig('jacobian.png')
```

Step-by-Step Debugging

```
ss.TDS.config.tf = 0.1  # Short simulation
ss.TDS.config.tstep = 0.001  # Small steps

# Enable verbose output
andes.config_logger(stream_level=10)  # DEBUG

ss.TDS.run()
```

Common Issues

Symptom	Possible Cause
Initialization fails	Wrong `v_str` or circular dependency
Flat run drifts	Equation sign error
NaN during simulation	Division by zero, missing parameter
Jacobian singular	Missing equation or wrong `e_str`
Results differ from PSS/E	Parameter unit mismatch

See Also

- *Model Structure* - Model structure reference
- *Verification* - Verification approach

CHAPTER

FIVE

REFERENCE

Technical reference documentation for ANDES, including command-line interface, configuration options, model specifications, and API documentation.

5.1 Command Line Interface

ANDES provides a command-line interface (CLI) for running simulations, plotting results, and managing the software without writing Python code. This reference documents all available commands and options.

5.1.1 Basic Usage

```
andes <command> [options] [arguments]
```

5.1.2 Commands Overview

Command	Purpose
`run`	Run power system simulations
`plot`	Plot time-domain simulation results
`doc`	View model and routine documentation
`prepare`	Generate numerical code from models
`selftest`	Verify installation
`misc`	Utility functions

5.1.3 andes run

Run power system simulations on case files.

Basic Examples

```
# Power flow (default routine)
andes run case.xlsx

# Time-domain simulation
andes run case.xlsx -r tds

# Eigenvalue analysis
andes run case.xlsx -r eig

# PSS/E files (RAW + DYR)
andes run system.raw --addfile system.dyr -r tds
```

Routine Options

Option	Routine	Description
(none)	PFlow	Newton-Raphson power flow
`-r tds`	TDS	Time-domain simulation
`-r eig`	EIG	Eigenvalue analysis
`-r pflow,tds`	Multiple	Run multiple routines

Simulation Control

```
# Set simulation end time
andes run case.xlsx -r tds --tf 30

# Set time step
andes run case.xlsx -r tds --tstep 0.01

# Configuration options
andes run case.xlsx -r tds -O TDS.tf=30 -O TDS.tstep=0.01
```

File Options

```
# Add dynamic data file
andes run case.raw --addfile case.dyr

# Multiple additional files
andes run case.raw -a models1.dyr -a models2.dyr

# Cross-format: RAW base with XLSX dynamics
andes run case.raw -a dynamics.xlsx

# Convert to XLSX format
andes run case.raw --convert

# Convert to JSON format
andes run case.raw --convert json

# Specify output directory
andes run case.xlsx -o results/

# Disable output files
andes run case.xlsx -n
```

Parallel Execution

```
# Run multiple cases (auto-parallel)
andes run *.xlsx -r tds

# Run with wildcard pattern
andes run kundur_*.xlsx -r tds

# Limit CPU usage
andes run *.xlsx -r tds --ncpu 4
```

Interactive Shell

Exit to IPython shell after simulation for interactive analysis:

```
andes run kundur_full.xlsx -r tds -s -n
```

The System object is available as `system`:

```
In [1]: system.GENROU.omega.v
Out[1]: array([1.0, 1.0, 1.0, 1.0])
```

Complete Options

Option	Description
`-r ROUTINE`	Routine to run (pflow, tds, eig)
`--tf TF`	Simulation end time [s]
`--tstep STEP`	Time step [s]
`-a, --addfile FILE`	Additional file(s); repeatable (e.g., `-a f1.dyr -a f2.xlsx`)
`-O KEY=VALUE`	Configuration option
`-o PATH`	Output directory
`-n, --no-output`	Disable file output
`-c, --convert`	Convert file format
`--ncpu N`	Number of CPUs
`-s, --shell`	Exit to IPython shell
`-b SHEET`	Add workbook sheet (e.g., Fault)

5.1.4 andes plot

Plot results from time-domain simulation.

Basic Usage

```
# Plot variable index 5 vs time (index 0)
andes plot case_out.lst 0 5

# Plot multiple variables
andes plot case_out.lst 0 5 6 7 8

# Plot range of variables
andes plot case_out.lst 0 2:21:6
```

Finding Variable Indices

The `.lst` file contains variable names and indices:

```
0, Time [s], $Time\ [s]$
1, delta GENROU 1, $\delta\ GENROU\ 1$
5, omega GENROU 1, $\omega\ GENROU\ 1$
```

Search by name:

```
# Find omega variables
andes plot case_out.lst --xargs "omega GENROU"
```

Plot Options

Option	Description
`--savefig`	Save figure to file
`--save FILE`	Save to specific filename
`-d`	Disable LaTeX rendering
`--to-csv`	Export data to CSV

5.1.5 andes doc

View documentation for models and routines.

```
# Model documentation
andes doc GENROU

# List all models
andes doc -l
andes doc --list

# Routine documentation
andes doc TDS

# Search models
andes doc --list | grep -i exciter
```

5.1.6 andes prepare

Generate numerical code from symbolic model definitions. Code generation happens automatically on first use; manual preparation is mainly needed during model development.

```
# Full code generation
andes prepare

# Force regeneration
andes prepare -f

# Incremental (only changed models)
andes prepare -i

# Quick mode (skip unchanged)
andes prepare -q
```

5.1.7 andes selftest

Verify installation by running the test suite.

```
# Full test
andes selftest

# Quick test (skip code generation)
andes selftest -q
```

5.1.8 andes misc

Utility functions.

```
# Show version
andes misc --version

# Edit configuration file
andes misc --edit-config

# Clean output files in current directory
andes misc -C

# Clean output files recursively
andes misc -C -r

# Save configuration to file
andes --save-config
```

5.1.9 Verbosity Levels

Control output detail with `-v LEVEL`:

Level	Name	Description
10	DEBUG	Detailed debugging info
20	INFO	Normal output (default)
30	WARNING	Warnings only
40	ERROR	Errors only

```
# Debug output
andes -v 10 run case.xlsx

# Quiet mode (warnings only)
andes -v 30 run case.xlsx
```

5.1.10 Environment Variables

Variable	Purpose
`ANDES_USE_UMFPACK`	Use UMFPACK sparse solver
`ANDES_DISABLE_NUMBA`	Disable Numba JIT compilation

5.1.11 Common Workflows

Standard Analysis

```
# Power flow only
andes run case.xlsx

# Power flow + TDS
andes run case.xlsx -r pflow,tds --tf 20

# All analyses
andes run case.xlsx -r pflow,tds,eig --tf 20
```

Batch Studies

```
# Run all cases in parallel
andes run cases/*.xlsx -r tds --ncpu 8

# Limit to 4 processes
andes run cases/*.xlsx -r tds --ncpu 4
```

Convert and Simulate

```
# Convert PSS/E to XLSX
andes run system.raw --addfile system.dyr --convert

# Run simulation on converted file
andes run system.xlsx -r tds --tf 20
```

Add Disturbance Sheets

```
# Add Fault sheet to workbook
andes run case.xlsx -b Fault

# Add multiple sheets
andes run case.xlsx -b Fault,Toggle,Alter
```

5.1.12 Getting Help

```
# General help
andes --help

# Command-specific help
andes run --help
andes plot --help
andes doc --help
```

5.1.13 See Also

- *Config Reference* - Configuration options
- *Tutorials* - Tutorials with examples

5.2 Configuration

ANDES behavior can be customized through configuration options for the system, routines, and models. This reference documents all available configuration options and how to modify them.

5.2.1 Configuration Resolution

ANDES resolves configuration from multiple sources using a layered priority system. When the same option is specified in more than one source, the higher-priority source takes precedence. The four sources, listed from lowest to highest priority, are:

1. **Compiled defaults**: Hard-coded values defined in each model, routine, and system configuration class. These values are used when no external source provides an override.
2. **User configuration file** (`andes.rc`): A persistent INI-format file located at `~/.andes/andes.rc`. Settings in this file apply globally to all ANDES sessions on the machine and serve as user-level defaults.
3. **Embedded case configuration** (`_config`): Configuration records embedded within xlsx or json case files. These settings travel with the case file and override any `andes.rc` values for the same option, ensuring reproducibility when cases are shared between users or machines. Note that PSS/E and MATPOWER formats do not support embedded configuration.

4. **Command-line overrides** (`config_option`): Options passed via the `-O` flag on the command line or the `config_option` argument in `andes.load()` / `andes.run()`. These have the highest priority and override all other sources.

The resolution is performed during system initialization in three phases. First, `andes.rc` is loaded into an internal configuration object. Second, if the case file contains a `_config` section, its entries are merged, overwriting any overlapping keys. Third, command-line overrides are applied. The merged configuration object is then distributed to the system, all routines, and all models during their construction.

This layered design allows a single case file to carry its intended simulation settings while still permitting per-user customization through the rc file and per-run adjustments through the command line.

```
Priority (lowest → highest):
compiled defaults  <  andes.rc  <  _config in data file  <  CLI config_option
```

5.2.2 Configuration Levels

ANDES uses a hierarchical configuration system with multiple levels, each controlling different aspects of the software.

Level	Scope	Example
System	Case-relevant global settings	`freq`, `mva`
Runtime	Machine/environment settings	`numba`, `sparselib`
Routine	Analysis settings	`PFlow.tol`, `TDS.tf`
Model	Model-specific	`TGOV1.allow_adjust`

System, routine, and model configurations are written to both `andes.rc` and `_config` in data files. Runtime configuration is written only to `andes.rc` (under the `[Runtime]` section) and is never embedded in case data files, because machine-specific settings such as JIT compilation or sparse solver selection should not travel with the case.

5.2.3 Viewing Configuration

All Configuration

After loading a system, you can inspect the configuration at each level by accessing the `config` attribute of the system, routines, or models.

```python
import andes

ss = andes.load('case.xlsx')

# System config (case-relevant)
print(ss.config)

# Runtime config (machine/environment)
```

(continues on next page)

(continued from previous page)

```
print(ss.runtime)

# Power flow config
print(ss.PFlow.config)

# TDS config
print(ss.TDS.config)

# Model config
print(ss.TGOV1.config)
```

Specific Options

Individual configuration values can be accessed directly using attribute notation, which is useful when you need to check or use a specific setting in your code.

```
# Get simulation end time
print(f"End time: {ss.TDS.config.tf}")

# Get convergence tolerance
print(f"Tolerance: {ss.PFlow.config.tol}")
```

5.2.4 Modifying Configuration

In Python

The most common approach is to modify configuration after loading the system but before running any analysis routines. This allows settings to be adjusted based on the specific requirements of each study.

```
ss = andes.load('case.xlsx')

# Modify before running
ss.TDS.config.tf = 20         # Simulation end time
ss.TDS.config.tstep = 0.01    # Time step
ss.PFlow.config.max_iter = 50  # Max iterations
```

At Load Time

Configuration options can also be passed when loading a system using the `config_option` argument. This is convenient when you want to set options in a single function call.

```
ss = andes.run('case.xlsx',
               config_option=['TDS.tf=20', 'PFlow.tol=1e-8'])
```

Command Line

When using the ANDES command-line interface, configuration options are specified using the `-O` flag followed by the option in `Section.option=value` format.

```
# Single option
andes run case.xlsx -r tds -O TDS.tf=20

# Multiple options
andes run case.xlsx -r tds -O TDS.tf=20 PFlow.max_iter=50
```

5.2.5 Configuration File

For persistent settings that apply across all ANDES sessions, you can create a configuration file that is automatically loaded when ANDES starts.

```
# Generate default config
andes --save-config

# Location: ~/.andes/andes.rc
```

Config File Format

The configuration file uses INI format with sections corresponding to the system, runtime, routines, and models. Options are specified as key-value pairs within each section.

```
[System]
freq = 60
mva = 100

[Runtime]
numba = 1
sparselib = klu
seed = 0

[PFlow]
tol = 1e-6
max_iter = 25

[TDS]
tf = 20
tstep = 0.0333
fixt = 1
max_iter = 15

[TGOV1]
allow_adjust = 1
```

The `[Runtime]` section contains machine-specific settings (JIT compilation, sparse solver selection, NumPy error behavior) that are not written to case data files. All other sections appear in both `andes.rc` and the `_config` embedded in xlsx/json data files.

5.2.6 System Options

System-level options control case-relevant global settings. These are written to data files and travel with the case.

Option	Default	Description
`freq`	60	System frequency [Hz]
`mva`	100	System base MVA
`diag_eps`	1e-8	Diagonal perturbation for Jacobian singularity
`warn_abnormal`	1	Warn on abnormal values during simulation

5.2.7 Runtime Options

Runtime options control machine-specific and environment settings. These are stored only in `andes.rc` under the `[Runtime]` section and are not embedded in case data files.

Option	Default	Description
`numba`	1	Enable Numba JIT compilation
`sparselib`	klu	Sparse solver library (`klu`, `umfpack`, `spsolve`, `cupy`)
`seed`	0	Random seed (0 = no seeding)
`np_divide`	warn	NumPy divide-by-zero behavior
`np_invalid`	warn	NumPy invalid floating-point behavior
`dime_enabled`	0	Enable DiME streaming
`save_stats`	0	Save call statistics

Runtime options can be accessed and modified via `ss.runtime`:

```
ss.runtime.numba = 1
ss.runtime.sparselib = 'umfpack'
```

5.2.8 Power Flow Options (PFlow)

Power flow configuration controls the Newton-Raphson iteration for steady-state solution.

Option	Default	Description
`tol`	1e-6	Convergence tolerance
`max_iter`	25	Maximum iterations

5.2.9 Time Domain Options (TDS)

Time-domain simulation options control the numerical integration and output behavior.

Option	Default	Description
`tf`	20	End time [s]
`tstep`	1/30	Time step [s]
`fixt`	1	Fixed time step (1) or adaptive (0)
`max_iter`	15	Max iterations per step
`tol`	1e-6	Convergence tolerance

5.2.10 Model Limit Options

All models with limiters support automatic limit adjustment. These options control whether and how limits can be adjusted when steady-state values exceed them.

Option	Default	Description
`allow_adjust`	1	Allow limit auto-adjustment
`adjust_lower`	0	Allow lowering lower limits
`adjust_upper`	1	Allow raising upper limits

5.2.11 Sparse Solvers

ANDES supports multiple sparse linear solvers for the Jacobian system. The choice of solver can affect both performance and numerical stability.

```
# Set solver via runtime config (system-wide)
ss.runtime.sparselib = 'klu'      # Default (fastest)
ss.runtime.sparselib = 'umfpack'  # Alternative
ss.runtime.sparselib = 'spsolve'  # Pure Python (slowest)
```

KLU is generally the fastest solver for power system problems and is recommended for most use cases.

5.2.12 Numba JIT Compilation

Numba provides just-in-time compilation of numerical functions for improved performance. When enabled, the first execution compiles the functions (which takes additional time), but subsequent executions are significantly faster.

```
# Enable via runtime config
ss = andes.load('case.xlsx')
ss.runtime.numba = True
```

5.2.13 Limit Adjustment

Models with limiters can automatically adjust their limits if steady-state values exceed the specified bounds. This feature helps simulations proceed even when data may have inconsistencies, but should be used with caution.

```
# Disable limit adjustment for a model
ss.TGOV1.config.allow_adjust = 0

# Allow only upper limit adjustment
ss.ESST1A.config.adjust_lower = 0
ss.ESST1A.config.adjust_upper = 1
```

Warning: Relying on automatic adjustment can mask underlying data issues. When limits are adjusted, ANDES reports the changes, and you should investigate whether the original parameters need correction.

5.2.14 Logging

Control the verbosity of ANDES output using the logging configuration. Higher levels produce less output, while lower levels provide more detail for debugging.

```
# Configure logging level
andes.config_logger(stream_level=20)  # INFO
andes.config_logger(stream_level=30)  # WARNING only
andes.config_logger(stream_level=10)  # DEBUG
```

Level	Name	Description
10	DEBUG	Detailed debugging info
20	INFO	Normal output (default)
30	WARNING	Warnings only
40	ERROR	Errors only

Command line:

```
andes run case.xlsx --verbose 10  # DEBUG
andes run case.xlsx --verbose 30  # WARNING
```

5.2.15 See Also

- *Command Line Interface* - Command line interface reference
- *Config Reference* - Auto-generated configuration reference

5.3 Model Reference

Use the navigation to browse models by group.

Supported Groups and Models

Group	Models
ACLine	*Line*
ACNode	*Bus*
ACShort	*Jumper*
Calculation	*ACE*, *ACEc*, *COI*
Collection	*Area*
DCLink	*Ground*, *R*, *L*, *C*, *RCp*, *RCs*, *RLs*, *RLCs*, *RLCp*
DCTopology	*Node*
DG	*PVD1*, *ESD1*, *EV1*, *EV2*
DGProtection	*DGPRCT1*, *DGPRCTExt*
DataSeries	*TimeSeries*
DynLoad	*ZIP*, *FLoad*
DynShaft	*SHAFT5*
Exciter	*EXDC2*, *IEEEX1*, *ESDC1A*, *ESDC2A*, *EXST1*, *ESST3A*, *SEXS*, *IEEET1*, *EXAC1*, *EXAC2*, *EXAC4*, *ESS*
FreqMeasurement	*BusFreq*, *BusROCOF*, *FreqDiv*
Information	*Summary*
Interface	*Fortescue*
Motor	*Motor3*, *Motor5*
OutputSelect	*Output*
PLL	*PLL1*, *PLL2*
PSS	*IEEEST*, *ST2CUT*
PhasorMeasurement	*PMU*
RenAerodynamics	*WTARA1*, *WTARV1*
RenExciter	*REECA1*, *REECA1E*, *REECA1G*, *REECB1*
RenGen	*REGCA1*, *REGCP1*, *REGCV1*, *REGCV2*, *REGF1*, *REGF2*, *REGF3*
RenGovernor	*WTDTA1*, *WTDS*
RenPitch	*WTPTA1*
RenPlant	*REPCA1*
RenTorque	*WTTQA1*
StaticACDC	*VSCShunt*
StaticGen	*PV*, *Slack*
StaticLoad	*PQ*
StaticShunt	*Shunt*, *ShuntTD*, *ShuntSw*
SynGen	*GENCLS*, *GENROU*, *PLBVFU1*
TimedEvent	*Toggle*, *Fault*, *Alter*
TurbineGov	*TG2*, *TGOV1*, *TGOV1DB*, *TGOV1N*, *TGOV1NDB*, *IEEEG1*, *IEESGO*, *GAST*, *HYGOV*, *HYGOVDB*, *H*
VoltComp	*IEEEVC*

5.3.1 ACLine

Common Parameters: u, name, bus1, bus2, r, x

Common Variables: v1, v2, a1, a2

Available models: *Line*

Line

AC transmission line model.

The model is also used for two-winding transformer. Transformers can set the tap ratio in `tap` and/or phase shift angle `phi`.

To reduce the number of variables, line injections are summed at bus equations and are not stored. Current injections are not computed.

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
bus1		idx of from bus			
bus2		idx of to bus			
Sn	S_n	Power rating	100	*MW*	non_zero
fn	f	rated frequency	60	*Hz*	
Vn1	V_{n1}	AC voltage rating	110	*kV*	non_zero
Vn2	V_{n2}	rated voltage of bus2	110	*kV*	non_zero
r	r	line resistance	0.000	*p.u.*	z
x	x	line reactance	0.000	*p.u.*	non_zero,z
b		shared shunt susceptance	0	*p.u.*	y
g		shared shunt conductance	0	*p.u.*	y
b1	b_1	from-side susceptance	0	*p.u.*	y
g1	g_1	from-side conductance	0	*p.u.*	y
b2	b_2	to-side susceptance	0	*p.u.*	y
g2	g_2	to-side conductance	0	*p.u.*	y
trans		transformer branch flag	0	*bool*	
tap	t_{ap}	transformer branch tap ratio	1	*float*	non_negative
phi	ϕ	transformer branch phase shift in rad	0	*radian*	
rate_a	R_{ATEA}	long-term flow limit (placeholder)	0	*MVA*	
rate_b	R_{ATEB}	short-term flow limit (placeholder)	0	*MVA*	
rate_c	R_{ATEC}	emergency flow limit (placeholder)	0	*MVA*	
owner		owner code			
xcoord		x coordinates			
ycoord		y coordinates			
ue	u_e	effective online status	0		

Variables

Name	Symbol	Type	Description	Unit	Properties
a1	a_1	ExtAlgeb	phase angle of the from bus		
a2	a_2	ExtAlgeb	phase angle of the to bus		
v1	v_1	ExtAlgeb	voltage magnitude of the from bus		
v2	v_2	ExtAlgeb	voltage magnitude of the to bus		

Initialization Equations

Name	Symbol	Type	Initial Value
a1	a_1	ExtAlgeb	
a2	a_2	ExtAlgeb	
v1	v_1	ExtAlgeb	
v2	v_2	ExtAlgeb	

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
a1	a_1	ExtAlgeb	$ue\left(-itapv_1v_2\left(-bhk\sin\left(-a_1+a_2+\phi\right)+ghk\cos\left(-a_1+a_2+\phi\right)\right)+itap_2v_1^2\left(gh+ghk\right)\right)$
a2	a_2	ExtAlgeb	$ue\left(-itapv_1v_2\left(bhk\sin\left(-a_1+a_2+\phi\right)+ghk\cos\left(-a_1+a_2+\phi\right)\right)+v_2^2\left(gh+ghk\right)\right)$
v1	v_1	ExtAlgeb	$ue\left(-itapv_1v_2\left(-bhk\cos\left(-a_1+a_2+\phi\right)-ghk\sin\left(-a_1+a_2+\phi\right)\right)-itap_2v_1^2\left(bh+bhk\right)\right)$
v2	v_2	ExtAlgeb	$ue\left(itapv_1v_2\left(bhk\cos\left(-a_1+a_2+\phi\right)-ghk\sin\left(-a_1+a_2+\phi\right)\right)-v_2^2\left(bh+bhk\right)\right)$

Services

Name	Symbol	Equation	Type
gh	g_h	$0.5g+g_1$	ConstService
bh	b_h	$0.5b+b_1$	ConstService
gk	g_k	$0.5g+g_2$	ConstService
bk	b_k	$0.5b+b_2$	ConstService
yh	y_h	$u\left(ibh+gh\right)$	ConstService
yk	y_k	$u\left(ibk+gk\right)$	ConstService
yhk	y_{hk}	$\frac{u}{r+i(x+1.0\cdot10^{-8})+1.0\cdot10^{-8}}$	ConstService
ghk	g_{hk}	$\mathrm{re}\left(yhk\right)$	ConstService
bhk	b_{hk}	$\mathrm{im}\left(yhk\right)$	ConstService
itap	$1/t_{ap}$	$\frac{1}{tap}$	ConstService
itap2	$1/t_{ap}^2$	$\frac{1}{tap^2}$	ConstService

Config Fields in [Line]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

5.3.2 ACNode

Common Parameters: u, name

Common Variables: a, v

Available models: *Bus*

Bus

AC Bus model.

Power balance equation have the form of `load - injection = 0`. Namely, load is positively summed, while injections are negative.

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
Vn	V_n	AC voltage rating	110	kV	non_zero
vmax	V_{max}	Voltage upper limit	1.100	*p.u.*	
vmin	V_{min}	Voltage lower limit	0.900	*p.u.*	
v0	V_0	initial voltage magnitude	1	*p.u.*	non_zero
a0	θ_0	initial voltage phase angle	0	*rad*	
xcoord		x coordinate (longitude)	0		
ycoord		y coordinate (latitude)	0		
area		Area code			
zone		Zone code			
owner		Owner code			
ue	u_e	effective online status	0		

Variables

Name	Symbol	Type	Description	Unit	Properties
a	a	Algeb	voltage angle	*rad*	v_str
v	v	Algeb	voltage magnitude	*p.u.*	v_str

Initialization Equations

Name	Symbol	Type	Initial Value
a	a	Algeb	$a_0\left(1 - z_{flat}\right) + 1.0 \cdot 10^{-8} z_{flat}$
v	v	Algeb	$v_0\left(1 - z_{flat}\right) + z_{flat}$

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
a	a	Algeb	0
v	v	Algeb	0

Config Fields in [Bus]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)
flat_start	z_{flat}	0	flat start for voltages	(0, 1)

5.3.3 ACShort

Common Parameters: u, name, bus1, bus2

Common Variables: v1, v2, a1, a2

Available models: *Jumper*

Jumper

Jumper is a device to short two buses (merging two buses into one).

Jumper can connect two buses satisfying one of the following conditions:

- neither bus is voltage-controlled
- either bus is voltage-controlled
- both buses are voltage-controlled, and the voltages are the same.

If the buses are controlled in different voltages, power flow will not solve (as the power flow through the jumper will be infinite).

In the solutions, the `p` and `q` are flowing out of bus1 and flowing into bus2.

Setting a Jumper's connectivity status `u` to zero will disconnect the two buses. In the case of a system split, one will need to call `System.connectivity()` immediately following the split to detect islands.

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
bus1		idx of from bus			
bus2		idx of to bus			
ue	u_e	effective online status	0		

Variables

Name	Symbol	Type	Description	Unit	Properties
p	p	Algeb	active power (1 to 2)		
q	q	Algeb	active power (1 to 2)		
a1	a_1	ExtAlgeb	phase angle of the from bus		
a2	a_2	ExtAlgeb	phase angle of the to bus		
v1	v_1	ExtAlgeb	voltage magnitude of the from bus		
v2	v_2	ExtAlgeb	voltage magnitude of the to bus		

Initialization Equations

Name	Symbol	Type	Initial Value
p	p	Algeb	
q	q	Algeb	
a1	a_1	ExtAlgeb	
a2	a_2	ExtAlgeb	
v1	v_1	ExtAlgeb	
v2	v_2	ExtAlgeb	

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
p	p	Algeb	$p\left(1-ue\right)+ue\left(a_1-a_2\right)$
q	q	Algeb	$p\left(1-ue\right)+ue\left(v_1-v_2\right)$
a1	a_1	ExtAlgeb	p
a2	a_2	ExtAlgeb	$-p$
v1	v_1	ExtAlgeb	q
v2	v_2	ExtAlgeb	$-q$

Config Fields in [Jumper]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

5.3.4 Calculation

Group of classes that calculates based on other models.

Common Parameters: u, name

Available models: *ACE*, *ACEc*, *COI*

ACE

Area Control Error model.

Discrete frequency sampling. System base frequency from `system.config.freq` is used.

Frequency sampling period (in seconds) can be specified in `ACE.config.interval`. The sampling start time (in seconds) can be specified in `ACE.config.offset`.

Note: area idx is automatically retrieved from *bus*.

Reference:

NERC, "Balancing and Frequency Control Reference Document, Chapter 1 Balancing Fundamentals: ACE Review", 2021. Available:

https://www.nerc.com/comm/RSTC_Reliability_Guidelines/Reference_Document_NERC_Balancing_and_Frequency_Control.pdf

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
bus		bus idx for freq. measurement			mandatory
bias	B	bias parameter	-1	*MW/0.1Hz*	non_positive,power
busf		Optional BusFreq device idx			
ue	u_e	effective online status	0		
area			0		

Variables

Name	Symbol	Type	Description	Unit	Properties
ace	ace	Algeb	area control error	*p.u. (MW)*	
f	f	ExtAlgeb	Bus frequency	*p.u. (Hz)*	

Initialization Equations

Name	Symbol	Type	Initial Value
ace	ace	Algeb	
f	f	ExtAlgeb	

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
ace	ace	Algeb	$-ace - 10bias f_{sys} imva \left(fs_v - 1\right)$
f	f	ExtAlgeb	0

Services

Name	Symbol	Equation	Type
imva	$1/S_{b,sys}$	$\frac{1}{S_{b,sys}}$	ConstService

Discretes

Name	Symbol	Type	Info
fs	f_s	Sampling	Sampled freq.

Config Fields in [ACE]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)
freq_model		BusFreq	default freq. measurement model	('BusFreq',)
interval		4	sampling time interval	
offset		0	sampling time offset	

ACEc

Area Control Error model, where bias (B) is an approximation of the actual frequency response (β). Given NERC's definition, the bias should be a negative number.

Continuous frequency sampling. System base frequency from `system.config.freq` is used.

Note: area idx is automatically retrieved from *bus*.

Reference:

NERC, "Balancing and Frequency Control Reference Document, Chapter 1 Balancing Fundamentals: Bias (B) vs. Frequency Response (Beta)", 2021. Available:

https://www.nerc.com/comm/RSTC_Reliability_Guidelines/Reference_Document_NERC_Balancing_and_Frequency_Control.pdf

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
bus		bus idx for freq. measurement			mandatory
bias	B	bias parameter	-1	*MW/0.1Hz*	non_positive,power
busf		Optional BusFreq device idx			
ue	u_e	effective online status	0		
area			0		

Variables

Name	Symbol	Type	Description	Unit	Properties
ace	ace	Algeb	area control error	*p.u. (MW)*	
f	f	ExtAlgeb	Bus frequency	*p.u. (Hz)*	

Initialization Equations

Name	Symbol	Type	Initial Value
ace	ace	Algeb	
f	f	ExtAlgeb	

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
ace	ace	Algeb	$-ace + 10 f_{sys} imva \left(f - 1\right) \left\|bias\right\|$
f	f	ExtAlgeb	0

Services

Name	Symbol	Equation	Type
imva	$1/S_{b,sys}$	$\frac{1}{S_{b,sys}}$	ConstService

Config Fields in [ACEc]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)
freq_model		BusFreq	default freq. measurement model	('BusFreq',)

COI

Center of inertia calculation class.

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
ue	u_e	effective online status	0		
M		Linearly stored SynGen.M	0		

Variables

Name	Symbol	Type	Description	Unit	Properties
wgen	$wgen$	ExtState	Linearly stored SynGen.omega		
agen	$agen$	ExtState	Linearly stored SynGen.delta		
omega	ω	Algeb	COI speed		v_str,v_setter
delta	δ	Algeb	COI rotor angle		v_str,v_setter
omega_sub	ω_{sub}	ExtAlgeb	COI frequency contribution of each generator		
delta_sub	δ_{sub}	ExtAlgeb	COI angle contribution of each generator		

Initialization Equations

Name	Symbol	Type	Initial Value
wgen	$wgen$	ExtState	
agen	$agen$	ExtState	
omega	ω	Algeb	$a0a$
delta	δ	Algeb	$d0a$
omega_sub	ω_{sub}	ExtAlgeb	
delta_sub	δ_{sub}	ExtAlgeb	

Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
wgen	$wgen$	ExtState	0	
agen	$agen$	ExtState	0	

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
omega	ω	Algeb	$-\omega$
delta	δ	Algeb	$-\delta$
omega_sub	ω_{sub}	ExtAlgeb	$Mwwgen$
delta_sub	δ_{sub}	ExtAlgeb	$Mwagen$

Services

Name	Symbol	Equation	Type
Mw	M_w	$\frac{M}{Mr}$	ConstService
d0w	$\delta_{gen,0,w}$	Mwd_0	ConstService
a0w	$\omega_{gen,0,w}$	Mwa_0	ConstService

Config Fields in [COI]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

5.3.5 Collection

Collection of topology models

Common Parameters: u, name

Available models: *Area*

Area

Area model.

Area collects back references from the Bus model and the ACNode group.

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
ue	u_e	effective online status	0		

Config Fields in [Area]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

5.3.6 DCLink

Basic DC links

Common Parameters: u, name

Available models: *Ground*, *R*, *L*, *C*, *RCp*, *RCs*, *RLs*, *RLCs*, *RLCp*

Ground

Ground model that sets the voltage of the connected DC node.

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
node		Node index			mandatory
voltage	V_0	Ground voltage (typically 0)	0	*p.u.*	
ue	u_e	effective online status	0		

Variables

Name	Symbol	Type	Description	Unit	Properties
Idc	Idc	Algeb	Ficticious current injection from ground		v_str
v	v	ExtAlgeb			

Initialization Equations

Name	Symbol	Type	Initial Value
Idc	Idc	Algeb	0
v	v	ExtAlgeb	

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
Idc	Idc	Algeb	$u\left(v - voltage\right)$
v	v	ExtAlgeb	$-Idc$

Config Fields in [Ground]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

R

Resistive dc line

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
node1		Node 1 index			mandatory
node2		Node 2 index			mandatory
Vdcn1	V_{dcn1}	DC voltage rating on node 1	100	kV	non_zero
Vdcn2	V_{dcn2}	DC voltage rating on node 2	100	kV	non_zero
Idcn	I_{dcn}	DC current rating	1	kA	non_zero
ue	u_e	effective online status	0		
R		DC line resistance	0.010	*p.u.*	non_zero,r

Variables

Name	Symbol	Type	Description	Unit	Properties
Idc	Idc	Algeb	Current from node 2 to 1	*p.u.*	v_str
v1	v_1	ExtAlgeb	DC voltage on node 1		
v2	v_2	ExtAlgeb	DC voltage on node 2		

Initialization Equations

Name	Symbol	Type	Initial Value
Idc	Idc	Algeb	$\frac{u(-v_1+v_2)}{R}$
v1	v_1	ExtAlgeb	
v2	v_2	ExtAlgeb	

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
Idc	Idc	Algeb	$-Idc + \frac{u(-v_1+v_2)}{R}$
v1	v_1	ExtAlgeb	$-Idc$
v2	v_2	ExtAlgeb	Idc

Config Fields in [R]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

L

Inductive dc line

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
node1		Node 1 index			mandatory
node2		Node 2 index			mandatory
Vdcn1	V_{dcn1}	DC voltage rating on node 1	100	kV	non_zero
Vdcn2	V_{dcn2}	DC voltage rating on node 2	100	kV	non_zero
Idcn	I_{dcn}	DC current rating	1	kA	non_zero
ue	u_e	effective online status	0		
L		DC line inductance	0.001	*p.u.*	non_zero,r

Variables

Name	Symbol	Type	Description	Unit	Properties
IL	IL	State	Inductance current	*p.u.*	v_str
v1	v_1	ExtAlgeb	DC voltage on node 1		
v2	v_2	ExtAlgeb	DC voltage on node 2		

Initialization Equations

Name	Symbol	Type	Initial Value
IL	IL	State	0
v1	v_1	ExtAlgeb	
v2	v_2	ExtAlgeb	

Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
IL	IL	State	$-u\left(v_1 - v_2\right)$	

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
v1	v_1	ExtAlgeb	$-IL$
v2	v_2	ExtAlgeb	IL

Config Fields in [L]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

C

Capacitive dc branch

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
node1		Node 1 index			mandatory
node2		Node 2 index			mandatory
Vdcn1	V_{dcn1}	DC voltage rating on node 1	100	*kV*	non_zero
Vdcn2	V_{dcn2}	DC voltage rating on node 2	100	*kV*	non_zero
Idcn	I_{dcn}	DC current rating	1	*kA*	non_zero
ue	u_e	effective online status	0		
C		DC capacitance	0.001	*p.u.*	non_zero,g

Variables

Name	Symbol	Type	Description	Unit	Properties
vC	vC	State	Capacitor current	*p.u.*	v_str
Idc	Idc	Algeb	Current from node 2 to 1	*p.u.*	v_str
v1	v_1	ExtAlgeb	DC voltage on node 1		
v2	v_2	ExtAlgeb	DC voltage on node 2		

Initialization Equations

Name	Symbol	Type	Initial Value
vC	vC	State	0
Idc	Idc	Algeb	0
v1	v_1	ExtAlgeb	
v2	v_2	ExtAlgeb	

Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
vC	vC	State	$-Idcu$	

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
Idc	Idc	Algeb	$Idc\left(1-u\right)+u\left(-v_1+v_2+vC\right)$
v1	v_1	ExtAlgeb	$-Idc$
v2	v_2	ExtAlgeb	Idc

Config Fields in [C]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

RCp

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
node1		Node 1 index			mandatory
node2		Node 2 index			mandatory
Vdcn1	V_{dcn1}	DC voltage rating on node 1	100	kV	non_zero
Vdcn2	V_{dcn2}	DC voltage rating on node 2	100	kV	non_zero
Idcn	I_{dcn}	DC current rating	1	kA	non_zero
ue	u_e	effective online status	0		
R	R	DC line resistance	0.010	*p.u.*	non_zero,r
C	C	DC capacitance	0.001	*p.u.*	non_zero,g

Variables

Name	Symbol	Type	Description	Unit	Properties
vC	vC	State	Capacitor current	*p.u.*	v_str
Idc	Idc	Algeb	Current from node 2 to 1	*p.u.*	v_str
v1	v_1	ExtAlgeb	DC voltage on node 1		
v2	v_2	ExtAlgeb	DC voltage on node 2		

Initialization Equations

Name	Symbol	Type	Initial Value
vC	vC	State	$v_1 - v_2$
Idc	Idc	Algeb	$\frac{-v_1+v_2}{R}$
v1	v_1	ExtAlgeb	
v2	v_2	ExtAlgeb	

Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
vC	vC	State	$-u\left(Idc - \frac{vC}{R}\right)$	C

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
Idc	Idc	Algeb	$Idc\,(1-u) + u\,(-v_1 + v_2 + vC)$
v1	v_1	ExtAlgeb	$-Idc$
v2	v_2	ExtAlgeb	Idc

Config Fields in [RCp]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

RCs

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
node1		Node 1 index			mandatory
node2		Node 2 index			mandatory
Vdcn1	V_{dcn1}	DC voltage rating on node 1	100	*kV*	non_zero
Vdcn2	V_{dcn2}	DC voltage rating on node 2	100	*kV*	non_zero
Idcn	I_{dcn}	DC current rating	1	*kA*	non_zero
ue	u_e	effective online status	0		
R	R	DC line resistance	0.010	*p.u.*	non_zero,r
C	C	DC capacitance	0.001	*p.u.*	non_zero,g

Variables

Name	Symbol	Type	Description	Unit	Properties
vC	vC	State	Capacitor current	*p.u.*	v_str
Idc	Idc	Algeb	Current from node 2 to 1	*p.u.*	v_str
v1	v_1	ExtAlgeb	DC voltage on node 1		
v2	v_2	ExtAlgeb	DC voltage on node 2		

Initialization Equations

Name	Symbol	Type	Initial Value
vC	vC	State	$v_1 - v_2$
Idc	Idc	Algeb	$\frac{-v_1+v_2}{R}$
v1	v_1	ExtAlgeb	
v2	v_2	ExtAlgeb	

Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
vC	vC	State	$-Idcu$	C

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
Idc	Idc	Algeb	$Idc\,(1-u) + u\,(-IdcR - v_1 + v_2 + vC)$
v1	v_1	ExtAlgeb	$-Idc$
v2	v_2	ExtAlgeb	Idc

Config Fields in [RCs]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

RLs

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
node1		Node 1 index			mandatory
node2		Node 2 index			mandatory
Vdcn1	V_{dcn1}	DC voltage rating on node 1	100	kV	non_zero
Vdcn2	V_{dcn2}	DC voltage rating on node 2	100	kV	non_zero
Idcn	I_{dcn}	DC current rating	1	kA	non_zero
ue	u_e	effective online status	0		
R	R	DC line resistance	0.010	*p.u.*	non_zero,r
L	L	DC line inductance	0.001	*p.u.*	non_zero,r

Variables

Name	Symbol	Type	Description	Unit	Properties
IL	IL	State	Inductance current	*p.u.*	v_str
Idc	Idc	Algeb	Current from node 2 to 1	*p.u.*	v_str
v1	v_1	ExtAlgeb	DC voltage on node 1		
v2	v_2	ExtAlgeb	DC voltage on node 2		

Initialization Equations

Name	Symbol	Type	Initial Value
IL	IL	State	$\frac{v_1 - v_2}{R}$
Idc	Idc	Algeb	$-\frac{u(v_1 - v_2)}{R}$
v1	v_1	ExtAlgeb	
v2	v_2	ExtAlgeb	

Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
IL	IL	State	$u\left(-ILR + v_1 - v_2\right)$	L

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
Idc	Idc	Algeb	$-ILu - Idc$
v1	v_1	ExtAlgeb	$-Idc$
v2	v_2	ExtAlgeb	Idc

Config Fields in [RLs]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

RLCs

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
node1		Node 1 index			mandatory
node2		Node 2 index			mandatory
Vdcn1	V_{dcn1}	DC voltage rating on node 1	100	kV	non_zero
Vdcn2	V_{dcn2}	DC voltage rating on node 2	100	kV	non_zero
Idcn	I_{dcn}	DC current rating	1	kA	non_zero
ue	u_e	effective online status	0		
R	R	DC line resistance	0.010	*p.u.*	non_zero,r
L	L	DC line inductance	0.001	*p.u.*	non_zero,r
C	C	DC capacitance	0.001	*p.u.*	non_zero,g

Variables

Name	Symbol	Type	Description	Unit	Properties
IL	IL	State	Inductance current	*p.u.*	v_str
vC	vC	State	Capacitor current	*p.u.*	v_str
Idc	Idc	Algeb	Current from node 2 to 1	*p.u.*	v_str
v1	v_1	ExtAlgeb	DC voltage on node 1		
v2	v_2	ExtAlgeb	DC voltage on node 2		

Initialization Equations

Name	Symbol	Type	Initial Value
IL	IL	State	0
vC	vC	State	$v_1 - v_2$
Idc	Idc	Algeb	0
v1	v_1	ExtAlgeb	
v2	v_2	ExtAlgeb	

Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
IL	IL	State	$u\left(-ILR + v_1 - v_2 - vC\right)$	L
vC	vC	State	ILu	C

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
Idc	Idc	Algeb	$-IL - Idc$
v1	v_1	ExtAlgeb	$-Idc$
v2	v_2	ExtAlgeb	Idc

Config Fields in [RLCs]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

RLCp

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
node1		Node 1 index			mandatory
node2		Node 2 index			mandatory
Vdcn1	V_{dcn1}	DC voltage rating on node 1	100	kV	non_zero
Vdcn2	V_{dcn2}	DC voltage rating on node 2	100	kV	non_zero
Idcn	I_{dcn}	DC current rating	1	kA	non_zero
ue	u_e	effective online status	0		
R	R	DC line resistance	0.010	*p.u.*	non_zero,r
L	L	DC line inductance	0.001	*p.u.*	non_zero,r
C	C	DC capacitance	0.001	*p.u.*	non_zero,g

Variables

Name	Symbol	Type	Description	Unit	Properties
IL	IL	State	Inductance current	*p.u.*	v_str
vC	vC	State	Capacitor current	*p.u.*	v_str
Idc	Idc	Algeb	Current from node 2 to 1	*p.u.*	v_str
v1	v_1	ExtAlgeb	DC voltage on node 1		
v2	v_2	ExtAlgeb	DC voltage on node 2		

Initialization Equations

Name	Symbol	Type	Initial Value
IL	IL	State	0
vC	vC	State	$v_1 - v_2$
Idc	Idc	Algeb	$\frac{-v_1+v_2}{R}$
v1	v_1	ExtAlgeb	
v2	v_2	ExtAlgeb	

Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
IL	IL	State	uvC	L
vC	vC	State	$-u\left(-IL + Idc - \frac{vC}{R}\right)$	C

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
Idc	Idc	Algeb	$Idc\,(1-u) + u\,(-v_1 + v_2 + vC)$
v1	v_1	ExtAlgeb	$-Idc$
v2	v_2	ExtAlgeb	Idc

Config Fields in [RLCp]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

5.3.7 DCTopology

Common Parameters: u, name

Common Variables: v

Available models: *Node*

Node

DC Node model.

A DC Node is like an AC Bus. DC devices need to be connected to Nodes to inject power flow.

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
Vdcn	V_{dcn}	DC voltage rating	100	*kV*	non_zero
Idcn	I_{dcn}	DC current rating	1	*kA*	non_zero
v0	V_{dc0}	initial voltage magnitude	1	*p.u.*	
xcoord		x coordinate (longitude)	0		
ycoord		y coordinate (latitude)	0		
area		Area code			
zone		Zone code			
owner		Owner code			
ue	u_e	effective online status	0		

Variables

Name	Symbol	Type	Description	Unit	Properties
v	v	Algeb	voltage magnitude	*p.u.*	v_str

Initialization Equations

Name	Symbol	Type	Initial Value
v	v	Algeb	$v_0 \left(1 - z_{flat}\right) + z_{flat}$

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
v	v	Algeb	0

Config Fields in [Node]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)
flat_start	z_{flat}	0	flat start for voltages	(0, 1)

5.3.8 DG

Distributed generation (small-scale).

See *SynGen* for the notes on replacing StaticGen and setting the power ratio parameters.

Common Parameters: u, name, bus, fn

Available models: *PVD1*, *ESD1*, *EV1*, *EV2*

PVD1

WECC Distributed PV model.

Device power rating is specified in *Sn*. Output currents are named *Ipout_y* and *Iqout_y*. Output power can be computed as `Pe = Ipout_y * v` and `Qe = Iqout_y * v`.

Frequency tripping response points *ft0*, *ft1*, *ft2*, and *ft3* must be monotinically increasing. Same rule applies to the voltage tripping response points *vt0*, *vt1*, *vt2*, and *vt3*. The program does not check these values, and the user is responsible for the parameter validity.

Frequency and voltage recovery latching is yet to be implemented.

The recommended approach for modifying setpoints is the group-level API, which works uniformly across all DG models:

```
ss.DG.set_pref(ss, dev_idx, value)     # writes to pref0
ss.DG.set_qref(ss, dev_idx, value)     # writes to qref0
ss.DG.set_paux(ss, dev_idx, value)     # writes to Pext0 (AGC signal)
```

For advanced usage, direct in-place array assignment is also supported (e.g., `ss.PVD1.pref0.v[:] = new_values`).

Maximum power limit *pmx* can be disabled by editing the configuration file by setting *plim=0*. It cannot be modified in runtime.

Reference: [1] ESIG, WECC Distributed and Small PV Plants Generic Model (PVD1), [Online], Available:

https://www.esig.energy/wiki-main-page/wecc-distributed-and-small-pv-plants-generic-model-pvd1/

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
bus		interface bus id			mandatory
gen		static generator index			mandatory
Sn	S_n	device MVA rating	100	*MVA*	
fn	f_n	nominal frequency	60	*Hz*	
busf		Optional BusFreq measurement device idx			
xc	x_c	coupling reactance	0	*p.u.*	z
pqflag		P/Q priority for I limit; 0-Q priority, 1-P priority		*bool*	mandatory
igreg		Remote bus idx for droop response, None for local			
qmx	q_{mx}	Max. reactive power command	0.330	*pu*	power
qmn	q_{mn}	Min. reactive power command	-0.330	*pu*	power
pmx	p_{mx}	maximum power limit	9999	*pu*	power
v0	v_0	Lower limit of deadband for Vdroop response	0.800	*pu*	non_zero
v1	v_1	Upper limit of deadband for Vdroop response	1.100	*pu*	non_zero
dqdv	dq/dv	Q-V droop characteristics (negative)	-1		non_zero,power
fdbd	f_{dbd}	frequency deviation deadband	-0.017	*Hz*	non_positive
ddn	D_{dn}	Gain after f deadband	0	*pu (MW)/Hz*	non_negative,powe
ialim	I_{alim}	Apparent power limit	1.300		non_zero,non_nega
vt0	V_{t0}	Voltage tripping response curve point 0	0.880	*p.u.*	non_zero,non_nega
vt1	V_{t1}	Voltage tripping response curve point 1	0.900	*p.u.*	non_zero,non_nega
vt2	V_{t2}	Voltage tripping response curve point 2	1.100	*p.u.*	non_zero,non_nega
vt3	V_{t3}	Voltage tripping response curve point 3	1.200	*p.u.*	non_zero,non_nega
vrflag	z_{VR}	V-trip is latching (0) or self-resetting (0-1)	0		
ft0	f_{t0}	Frequency tripping response curve point 0	59.500	*Hz*	non_zero,non_nega
ft1	f_{t1}	Frequency tripping response curve point 1	59.700	*Hz*	non_zero,non_nega
ft2	f_{t2}	Frequency tripping response curve point 2	60.300	*Hz*	non_zero,non_nega
ft3	f_{t3}	Frequency tripping response curve point 3	60.500	*Hz*	non_zero,non_nega
frflag	z_{FR}	f-trip is latching (0) or self-resetting (0-1)	0		
tip	T_{ip}	Inverter active current lag time constant	0.020	*s*	non_negative
tiq	T_{iq}	Inverter reactive current lag time constant	0.020	*s*	non_negative
gammap	γ_p	Ratio of PVD1.pref0 w.r.t to that of static PV	1		
gammaq	γ_q	Ratio of PVD1.qref0 w.r.t to that of static PV	1		
recflag	z_{rec}	Enable flag for voltage and frequency recovery limiters	1		

continues on

Table 2 – continued from previous page

Name	Symbol	Description	Default	Unit	Properties
ue	u_e	effective online status	0		

Variables

Name	Symbol	Type	Description	Unit	Properties
Ipout_y	$Ipout_y$	State	State in lag transfer function		v_str
Iqout_y	$Iqout_y$	State	State in lag transfer function		v_str
fHz	fHz	Algeb	frequency in Hz	*Hz*	v_str
Ffl	Ffl	Algeb	Coeff. for under frequency		v_str
Ffh	Ffh	Algeb	Coeff. for over frequency		v_str
Fdev	$Fdev$	Algeb	Frequency deviation	*Hz*	v_str
DB_y	DB_y	Algeb	Deadband type 1 output		v_str
Fvl	Fvl	Algeb	Coeff. for under voltage		v_str
Fvh	Fvh	Algeb	Coeff. for over voltage		v_str
vp	vp	Algeb	Sensed positive voltage		v_str
Pext	$Pext$	Algeb	External power signal (for AGC)		v_str
Pref	$Pref$	Algeb	Reference power signal (for scheduling setpoint)		v_str
Psum	$Psum$	Algeb	Sum of P signals		v_str
Qdrp	$Qdrp$	Algeb	External power signal (for AGC)		v_str
Qref	$Qref$	Algeb	Reference power signal (for scheduling setpoint)		v_str
Qsum	$Qsum$	Algeb	Sum of Q signals		v_str
Ipul	$Ipul$	Algeb	Ipcmd before Ip hard limit		v_str
Iqul	$Iqul$	Algeb	Iqcmd before Iq hard limit		v_str
Ipmax	$Ipmax$	Algeb	Upper limit of Ip		v_str
Iqmax	$Iqmax$	Algeb	Upper limit of Iq		v_str
Ipcmd_x	$Ipcmd_x$	Algeb	Value before limiter		v_str
Iqcmd_x	$Iqcmd_x$	Algeb	Value before limiter		v_str
a	a	ExtAlgeb	bus (or igreg) phase angle	*rad.*	
v	v	ExtAlgeb	bus (or igreg) terminal voltage	*p.u.*	
f	f	ExtAlgeb	Bus frequency	*p.u.*	

Initialization Equations

Nam	Sym bol	Type	Initial Value
Ipout	$Ipou$	State	$1.0Ipcmd_{limzi}Ipcmd_x\left(FfhFflFvhFvlrecflag - recflag + 1\right) + 1.0Ipcmd_{limzu}Ipmax\left(FfhFflFvhFvlrecflag - recflag + 1\right)$
Iqout	$Iqou$	State	$1.0Iqcmd_{limzi}Iqcmd_x\left(FfhFflFvhFvlrecflag - recflag + 1\right) - 1.0Iqcmd_{limzl}Iqmax\left(FfhFflFvhFvlrecflag - recflag + 1\right) + 1.0Iqcmd_{limzu}Iqmax\left(FfhFflFvhFvlrecflag - recflag + 1\right)$
fHz	fHz	Algeb	ffn
Ffl	Ffl	Algeb	$FL_{1zi}Kft_{01}\left(fHz - ft_0\right) + FL_{1zu}$
Ffh	Ffh	Algeb	$FL_{2zi}\left(Kft_{23}\left(-fHz + ft_2\right) + 1\right) + FL_{2zl}$
Fdev	$Fdev$	Algeb	$-fHz + fn$
DB_y	DB_y	Algeb	$ddn\left(DB_{dbzl}\left(Fdev - fdbd\right) + DB_{dbzu}\left(Fdev - 999.0\right)\right)$
Fvl	Fvl	Algeb	$Kvt_{01}VL_{1zi}\left(v - vt_0\right) + VL_{1zu}$
Fvh	Fvh	Algeb	$VL_{2zi}\left(Kvt_{23}\left(-v + vt_2\right) + 1\right) + VL_{2zl}$
vp	vp	Algeb	$VLo_{zi}v + 0.01VLo_{zl}$
Pext	$Pext$	Algeb	$Pext_0u$
Pref	$Pref$	Algeb	$pref_0u$
Psum	$Psum$	Algeb	$u\left(DB_y + Pext + Pref\right)$
Qdrp	$Qdrp$	Algeb	$VQ_{1zi}ue\left(dqdv\left(-Vcomp + Vqu\right) + qmx\right) + VQ_{1zl}qmxue + VQ_{2zi}dqdvue\left(-Vcomp + v_1\right) + VQ_{2zu}qmn$
Qref	$Qref$	Algeb	$qref_0u$
Qsum	$Qsum$	Algeb	$u\left(VQ_{1zi}ue\left(dqdv\left(-Vcomp + Vqu\right) + qmx\right) + VQ_{1zl}qmxue + VQ_{2zi}dqdvue\left(-Vcomp + v_1\right) + V$
Ipul	$Ipul$	Algeb	$\frac{PHL_{zi}Psum + PHL_{zu}pmx}{vp}$
Iqul	$Iqul$	Algeb	$\frac{Qsum}{vp}$
Ipmax	$Ipmax$	Algeb	$\sqrt{Ipmaxsq_0}SWPQ_{s0} + SWPQ_{s1}ialim$
Iqmax	$Iqmax$	Algeb	$\sqrt{Iqmaxsq_0}SWPQ_{s1} + SWPQ_{s0}ialim$
Ipcm	$Ipcm$	Algeb	$Ipul$
Iqcm	$Iqcm$	Algeb	$Iqul$
a	a	ExtAlgeb	

Differential Equations

Nam	Sym bol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
Ipout	$Ipou$	State	$1.0Ipcmd_{limzi}Ipcmd_x\left(FfhFflFvhFvlrecflag - recflag + 1\right) + 1.0Ipcmd_{limzu}Ipmax\left(FfhFflFvhFvlrecflag - recflag + 1\right) - Ipout_y$	T_{ip}
Iqout	$Iqou$	State	$1.0Iqcmd_{limzi}Iqcmd_x\left(FfhFflFvhFvlrecflag - recflag + 1\right) - 1.0Iqcmd_{limzl}Iqmax\left(FfhFflFvhFvlrecflag - recflag + 1\right) + 1.0Iqcmd_{limzu}Iqmax\left(FfhFflFvhFvlrecflag - recflag + 1\right) - Iqout_y$	T_{iq}

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
fHz	fHz	Algeb	$ffn - fHz$
Ffl	Ffl	Algeb	$FL_{1zi}Kft_{01}\left(fHz - ft_0\right) + FL_{1zu} - Ffl$
Ffh	Ffh	Algeb	$FL_{2zi}\left(Kft_{23}\left(-fHz + ft_2\right) + 1\right) + FL_{2zl} - Ffh$
Fdev	$Fdev$	Algeb	$-Fdev - fHz + fn$
DB_y	DB_y	Algeb	$-DB_y + ddn\left(DB_{dbzl}\left(Fdev - fdbd\right) + DB_{dbzu}\left(Fdev - 999.0\right)\right)$
Fvl	Fvl	Algeb	$-Fvl + Kvt_{01}VL_{1zi}\left(v - vt_0\right) + VL_{1zu}$
Fvh	Fvh	Algeb	$-Fvh + VL_{2zi}\left(Kvt_{23}\left(-v + vt_2\right) + 1\right) + VL_{2zl}$
vp	vp	Algeb	$VLo_{zi}v + 0.01VLo_{zl} - vp$
Pext	$Pext$	Algeb	$-Pext + Pext_0ue$
Pref	$Pref$	Algeb	$-Pref + pref_0ue$
Psum	$Psum$	Algeb	$-Psum + ue\left(DB_y + Pext + Pref\right)$
Qdrp	$Qdrp$	Algeb	$-Qdrp + VQ_{1zi}ue\left(dqdv\left(-Vcomp + Vqu\right) + qmx\right) + VQ_{1zl}qmxue + VQ_{2zi}dqdvue\left(-Vcomp + v_1\right) + VQ_{2zu}qmn$
Qref	$Qref$	Algeb	$-Qref + qref_0ue$
Qsum	$Qsum$	Algeb	$-Qsum + ue\left(Qdrp + Qref\right)$
Ipul	$Ipul$	Algeb	$-Ipul + \frac{PHL_{zi}Psum + PHL_{zu}pmx}{vp}$
Iqul	$Iqul$	Algeb	$-Iqul + \frac{Qsum}{vp}$
Ipmax	$Ipmax$	Algeb	$-Ipmax + \sqrt{Ipmaxsq}SWPQ_{s0} + SWPQ_{s1}ialim$
Iqmax	$Iqmax$	Algeb	$-Iqmax + \sqrt{Iqmaxsq}SWPQ_{s1} + SWPQ_{s0}ialim$
Ipcmd_	$Ipcmd$	Algeb	$-Ipcmd_x + Ipul$
Iqcmd_	$Iqcmd$	Algeb	$-Iqcmd_x + Iqul$
a	a	ExtAlgeb	$-Ipout_yuev$
v	v	ExtAlgeb	$-Iqout_yuev$
f	f	ExtAlgeb	0

Services

Nam	Symbol	Equation	Type
pref0	P_{ref0}	$gammapp0s$	ConstService
qref0	Q_{ref0}	$gammaqq0s$	ConstService
Kft0	K_{ft0}	$\frac{1}{-ft_0+ft_1}$	ConstService
Kft2	K_{ft2}	$\frac{1}{-ft_2+ft_3}$	ConstService
Kvt0	K_{vt0}	$\frac{1}{-vt_0+vt_1}$	ConstService
Kvt2	K_{vt2}	$\frac{1}{-vt_2+vt_3}$	ConstService
Pext0	P_{ext0}	0	ConstService
Vcon	V_{com}	$\sqrt{Ipout_y^2xc^2 - iIpout_yvxce^{ia} + iIpout_yvxce^{-ia} + Iqout_y^2xc^2 - Iqout_yvxce^{ia} - I}$	VarService $vxce^{-ia} + v^2$
Vqu	V_{qu}	$v_1 - \frac{-qmn+qref_0}{dqdv}$	ConstService
Vql	V_{ql}	$v_0 + \frac{qmx-qref_0}{dqdv}$	ConstService
Ipmaxs	I_{pma}^2	FixPiecewise $\left(\left(0,\ ialim^2 - (Iqcmd_{limzi}Iqcmd_x\left(FfhFflFvhFvlrecflag - recf\right.\right.\right.$	VarService $1) - Iqcmd_{limz}$
Ipmaxs	I_{pma}^2	FixPiecewise $\left(\left(0,\ ialim^2 - \frac{qref_0^2u^2}{v^2} \leq 0\right), \left(ialim^2 - \frac{qref_0^2u^2}{v^2},\ \text{True}\right)\right)$	ConstService
Iqmaxs	I_{qma}^2	FixPiecewise $\left(\left(0,\ ialim^2 - (Ipcmd_{limzi}Ipcmd_x\left(FfhFflFvhFvlrecflag - recf\right.\right.\right.$	VarService $1) + Ipcmd_{limz}$

Discretes

Name	Symbol	Type	Info
SWPQ	SW_{PQ}	Switcher	
FL1	$FL1$	Limiter	Under frequency comparer
FL2	$FL2$	Limiter	Over frequency comparer
DB_db	db_{DB}	DeadBand	
VL1	$VL1$	Limiter	Under voltage comparer
VL2	$VL2$	Limiter	Over voltage comparer
VLo	VLo	Limiter	Voltage lower limit (0.01) flag
PHL	PHL	Limiter	limiter for Psum in [0, pmx]
VQ1	$VQ1$	Limiter	Under voltage comparer for Q droop
VQ2	$VQ2$	Limiter	Over voltage comparer for Q droop
Ipcmd_lim	lim_{Ipcmd}	HardLimiter	
Iqcmd_lim	lim_{Iqcmd}	HardLimiter	

Blocks

Name	Symbol	Type	Info
DB	DB	DeadBand1	frequency deviation deadband with gain
Ipcmd	I^{pcmd}	GainLimiter	Ip with limiter and coeff.
Iqcmd	I^{qcmd}	GainLimiter	Iq with limiter and coeff.
Ipout	$Ipout$	Lag	Output Ip filter
Iqout	$Iqout$	Lag	Output Iq filter

Setpoints

Name	Target	Info
pref	pref0	Initial P for the PVD1 device
qref	qref0	Initial Q for the PVD1 device
paux	Pext0	External additional signal added to Pext

Config Fields in [PVD1]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)
plim	P_{lim}	1	enable input power limit check bound by [0, pmx]	(0, 1)

ESD1

Distributed energy storage model.

A state-of-charge limit is added to the PVD1 model. This limit is applied to Ipmax and Ipmin. The state of charge is in state variable `SOC`, which is an alias of `pIG_y`.

Reference: [1] Powerworld, Renewable Energy Electrical Control Model REEC_C Available:

https://www.powerworld.com/WebHelp/Content/TransientModels_HTML/Exciter%20REEC_C.htm

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
bus		interface bus id			mandatory
gen		static generator index			mandatory
Sn	S_n	device MVA rating	100	*MVA*	
fn	f_n	nominal frequency	60	*Hz*	
busf		Optional BusFreq measurement device idx			
xc	x_c	coupling reactance	0	*p.u.*	z
pqflag		P/Q priority for I limit; 0-Q priority, 1-P priority		*bool*	mandatory
igreg		Remote bus idx for droop response, None for local			
qmx	q_{mx}	Max. reactive power command	0.330	*pu*	power
qmn	q_{mn}	Min. reactive power command	-0.330	*pu*	power
pmx	p_{mx}	maximum power limit	9999	*pu*	power
v0	v_0	Lower limit of deadband for Vdroop response	0.800	*pu*	non_zero
v1	v_1	Upper limit of deadband for Vdroop response	1.100	*pu*	non_zero
dqdv	dq/dv	Q-V droop characteristics (negative)	-1		non_zero,power
fdbd	f_{dbd}	frequency deviation deadband	-0.017	*Hz*	non_positive
ddn	D_{dn}	Gain after f deadband	0	*pu (MW)/Hz*	non_negative,pov
ialim	I_{alim}	Apparent power limit	1.300		non_zero,non_ne
vt0	V_{t0}	Voltage tripping response curve point 0	0.880	*p.u.*	non_zero,non_ne

continues o

Table 3 – continued from previous page

Name	Symbol	Description	Default	Unit	Properties
vt1	V_{t1}	Voltage tripping response curve point 1	0.900	*p.u.*	non_zero,non_ne
vt2	V_{t2}	Voltage tripping response curve point 2	1.100	*p.u.*	non_zero,non_ne
vt3	V_{t3}	Voltage tripping response curve point 3	1.200	*p.u.*	non_zero,non_ne
vrflag	z_{VR}	V-trip is latching (0) or self-resetting (0-1)	0		
ft0	f_{t0}	Frequency tripping response curve point 0	59.500	*Hz*	non_zero,non_ne
ft1	f_{t1}	Frequency tripping response curve point 1	59.700	*Hz*	non_zero,non_ne
ft2	f_{t2}	Frequency tripping response curve point 2	60.300	*Hz*	non_zero,non_ne
ft3	f_{t3}	Frequency tripping response curve point 3	60.500	*Hz*	non_zero,non_ne
frflag	z_{FR}	f-trip is latching (0) or self-resetting (0-1)	0		
tip	T_{ip}	Inverter active current lag time constant	0.020	*s*	non_negative
tiq	T_{iq}	Inverter reactive current lag time constant	0.020	*s*	non_negative
gammap	γ_p	Ratio of PVD1.pref0 w.r.t to that of static PV	1		
gammaq	γ_q	Ratio of PVD1.qref0 w.r.t to that of static PV	1		
recflag	z_{rec}	Enable flag for voltage and frequency recovery limiters	1		
Tf	T_f	Integrator constant for SOC model	1		
SOCmin	SOC_{min}	Minimum required value for SOC in limiter	0		
SOCmax	SOC_{max}	Maximum allowed value for SOC in limiter	1		
SOCinit	SOC_{init}	Initial state of charge	0.500		
En	E_n	Rated energy capacity	100	*MWh*	
EtaC	Eta_C	Efficiency during charging	1		
EtaD	Eta_D	Efficiency during discharging	1		
ue	u_e	effective online status	0		

Variables

Name	Symbol	Type	Description	Unit	Properties
Ipout_y	$Ipout_y$	State	State in lag transfer function		v_str
Iqout_y	$Iqout_y$	State	State in lag transfer function		v_str
pIG_y	pIG_y	State	Integrator output		v_str
SOC	SOC	AliasState	Alias for state of charge		
fHz	fHz	Algeb	frequency in Hz	*Hz*	v_str
Ffl	Ffl	Algeb	Coeff. for under frequency		v_str
Ffh	Ffh	Algeb	Coeff. for over frequency		v_str
Fdev	$Fdev$	Algeb	Frequency deviation	*Hz*	v_str
DB_y	DB_y	Algeb	Deadband type 1 output		v_str
Fvl	Fvl	Algeb	Coeff. for under voltage		v_str
Fvh	Fvh	Algeb	Coeff. for over voltage		v_str
vp	vp	Algeb	Sensed positive voltage		v_str
Pext	$Pext$	Algeb	External power signal (for AGC)		v_str
Pref	$Pref$	Algeb	Reference power signal (for scheduling setpoint)		v_str
Psum	$Psum$	Algeb	Sum of P signals		v_str
Qdrp	$Qdrp$	Algeb	External power signal (for AGC)		v_str
Qref	$Qref$	Algeb	Reference power signal (for scheduling setpoint)		v_str
Qsum	$Qsum$	Algeb	Sum of Q signals		v_str
Ipul	$Ipul$	Algeb	Ipcmd before Ip hard limit		v_str
Iqul	$Iqul$	Algeb	Iqcmd before Iq hard limit		v_str
Ipmax	$Ipmax$	Algeb	Upper limit of Ip		v_str
Iqmax	$Iqmax$	Algeb	Upper limit of Iq		v_str
Ipcmd_x	$Ipcmd_x$	Algeb	Value before limiter		v_str
Iqcmd_x	$Iqcmd_x$	Algeb	Value before limiter		v_str
Ipmin	$Ipmin$	Algeb	Minimum value of Ip		v_str
a	a	ExtAlgeb	bus (or igreg) phase angle	*rad.*	
v	v	ExtAlgeb	bus (or igreg) terminal voltage	*p.u.*	
f	f	ExtAlgeb	Bus frequency	*p.u.*	

Initialization Equations

Nam	Sym bol	Type	Initial Value
Ipout	$Ipou$	State	$1.0Ipcmd_{limzi}Ipcmd_x\left(FfhFflFvhFvlrecflag - recflag + 1\right) + 1.0Ipcmd_{limzu}Ipmax\left(FfhFflFvhFvlrecflag - recflag + 1\right)$
Iqout	$Iqou$	State	$1.0Iqcmd_{limzi}Iqcmd_x\left(FfhFflFvhFvlrecflag - recflag + 1\right) - 1.0Iqcmd_{limzl}Iqmax\left(FfhFflFvhFvlrecflag - recflag + 1\right) + 1.0Iqcmd_{limzu}Iqmax\left(FfhFflFvhFvlrecflag - recflag + 1\right)$
pIG_	$pIG_{\imath}$	State	$SOCinit$
SOC	SOC	Alias	
fHz	fHz	Algeb	ffn
Ffl	Ffl	Algeb	$FL_{1zi}Kft_{01}\left(fHz - ft_0\right) + FL_{1zu}$
Ffh	Ffh	Algeb	$FL_{2zi}\left(Kft_{23}\left(-fHz + ft_2\right) + 1\right) + FL_{2zl}$
Fdev	$Fdev$	Algeb	$-fHz + fn$
DB_y	DB_y	Algeb	$ddn\left(DB_{dbzl}\left(Fdev - fdbd\right) + DB_{dbzu}\left(Fdev - 999.0\right)\right)$
Fvl	Fvl	Algeb	$Kvt_{01}VL_{1zi}\left(v - vt_0\right) + VL_{1zu}$
Fvh	Fvh	Algeb	$VL_{2zi}\left(Kvt_{23}\left(-v + vt_2\right) + 1\right) + VL_{2zl}$
vp	vp	Algeb	$VLo_{zi}v + 0.01VLo_{zl}$
Pext	$Pext$	Algeb	$Pext_0u$
Pref	$Pref$	Algeb	$pref_0u$
Psum	$Psum$	Algeb	$u\left(DB_y + Pext + Pref\right)$
Qdrp	$Qdrp$	Algeb	$VQ_{1zi}ue\left(dqdv\left(-Vcomp + Vqu\right) + qmx\right) + VQ_{1zl}qmxue + VQ_{2zi}dqdvue\left(-Vcomp + v_1\right) + VQ_{2zu}qmn$
Qref	$Qref$	Algeb	$qref_0u$
Qsum	$Qsum$	Algeb	$u\left(VQ_{1zi}ue\left(dqdv\left(-Vcomp + Vqu\right) + qmx\right) + VQ_{1zl}qmxue + VQ_{2zi}dqdvue\left(-Vcomp + v_1\right) +$
Ipul	$Ipul$	Algeb	$\frac{PHL_{zi}Psum + PHL_{zu}pmx}{vp}$
Iqul	$Iqul$	Algeb	$\frac{Qsum}{vp}$
Ipmax	$Ipmax$	Algeb	$\left(1 - SOClim_{zl}\right)\left(\sqrt{Ipmaxsq_0}SWPQ_{s0} + SWPQ_{s1}ialim\right)$
Iqmax	$Iqmax$	Algeb	$\sqrt{Iqmaxsq_0}SWPQ_{s1} + SWPQ_{s0}ialim$
Ipcm	$Ipcm$	Algeb	$Ipul$
Iqcm	$Iqcm$	Algeb	$Iqul$

Differential Equations

Nam	Sym bol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
Ipout	$Ipou$	State	$1.0Ipcmd_{limzi}Ipcmd_x\left(FfhFflFvhFvlrecflag - recflag + 1\right) + 1.0Ipcmd_{limzu}Ipmax\left(FfhFflFvhFvlrecflag - recflag + 1\right) - Ipout_y$	T_{ip}
Iqout	$Iqou$	State	$1.0Iqcmd_{limzi}Iqcmd_x\left(FfhFflFvhFvlrecflag - recflag + 1\right) - 1.0Iqcmd_{limzl}Iqmax\left(FfhFflFvhFvlrecflag - recflag + 1\right) + 1.0Iqcmd_{limzu}Iqmax\left(FfhFflFvhFvlrecflag - recflag + 1\right) - Iqout_y$	T_{iq}
pIG_	$pIG_{\imath}$	State	$\frac{S_{b,sys}\left(-EtaCIpout_yLTN_{z1}v - \frac{Ipout_yLTN_{z0}v}{EtaD}\right)}{3600En}$	T_f
SOC	SOC	Alias	0	

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
fHz	fHz	Algeb	$ffn - fHz$
Ffl	Ffl	Algeb	$FL_{1zi}Kft_{01}\left(fHz - ft_0\right) + FL_{1zu} - Ffl$
Ffh	Ffh	Algeb	$FL_{2zi}\left(Kft_{23}\left(-fHz + ft_2\right) + 1\right) + FL_{2zl} - Ffh$
Fdev	$Fdev$	Algeb	$-Fdev - fHz + fn$
DB_y	DB_y	Algeb	$-DB_y + ddn\left(DB_{dbzl}\left(Fdev - fdbd\right) + DB_{dbzu}\left(Fdev - 999.0\right)\right)$
Fvl	Fvl	Algeb	$-Fvl + Kvt_{01}VL_{1zi}\left(v - vt_0\right) + VL_{1zu}$
Fvh	Fvh	Algeb	$-Fvh + VL_{2zi}\left(Kvt_{23}\left(-v + vt_2\right) + 1\right) + VL_{2zl}$
vp	vp	Algeb	$VLo_{zi}v + 0.01VLo_{zl} - vp$
Pext	$Pext$	Algeb	$-Pext + Pext_0ue$
Pref	$Pref$	Algeb	$-Pref + pref_0ue$
Psum	$Psum$	Algeb	$-Psum + ue\left(DB_y + Pext + Pref\right)$
Qdrp	$Qdrp$	Algeb	$-Qdrp + VQ_{1zi}ue\left(dqdv\left(-Vcomp + Vqu\right) + qmx\right) + VQ_{1zl}qmxue + VQ_{2zi}dqdvue\left(-Vcomp + v_1\right) + VQ_{2zu}qmn$
Qref	$Qref$	Algeb	$-Qref + qref_0ue$
Qsum	$Qsum$	Algeb	$-Qsum + ue\left(Qdrp + Qref\right)$
Ipul	$Ipul$	Algeb	$-Ipul + \frac{PHL_{zi}Psum + PHL_{zu}pmx}{vp}$
Iqul	$Iqul$	Algeb	$-Iqul + \frac{Qsum}{vp}$
Ipmax	$Ipmax$	Algeb	$-Ipmax + \left(1 - SOClim_{zl}\right)\left(\sqrt{Ipmaxsq}SWPQ_{s0} + SWPQ_{s1}ialim\right)$
Iqmax	$Iqmax$	Algeb	$-Iqmax + \sqrt{Iqmaxsq}SWPQ_{s1} + SWPQ_{s0}ialim$
Ipcmd_	$Ipcmd$	Algeb	$-Ipcmd_x + Ipul$
Iqcmd_	$Iqcmd$	Algeb	$-Iqcmd_x + Iqul$
Ipmin	$Ipmin$	Algeb	$-Ipmin + \left(SOClim_{zu} - 1\right)\left(\sqrt{Ipmaxsq}SWPQ_{s0} + SWPQ_{s1}ialim\right)$
a	a	ExtAlgeb	$-Ipout_yuev$
v	v	ExtAlgeb	$-Iqout_yuev$
f	f	ExtAlgeb	0

Services

Nam	Sym bol	Equation	Type
pref0	P_{ref0}	$gammapp0s$	Const-Service
qref0	Q_{ref0}	$gammaqq0s$	Const-Service
Kft0	K_{ft0}	$\frac{1}{-ft_0+ft_1}$	Const-Service
Kft2	K_{ft2}	$\frac{1}{-ft_2+ft_3}$	Const-Service
Kvt0	K_{vt0}	$\frac{1}{-vt_0+vt_1}$	Const-Service
Kvt2	K_{vt2}	$\frac{1}{-vt_2+vt_3}$	Const-Service
Pext0	P_{ext0}	0	Const-Service
Vcon	V_{com}	$\sqrt{Ipout_y^2xc^2 - iIpout_yvxce^{ia} + iIpout_yvxce^{-ia} + Iqout_y^2xc^2 - Iqout_yvxce^{ia} - I}$	VarService
Vqu	V_{qu}	$v_1 - \frac{-qmn+qref_0}{dqdv}$	Const-Service
Vql	V_{ql}	$v_0 + \frac{qmx-qref_0}{dqdv}$	Const-Service
Ip-maxs	I_{pma}^2	FixPiecewise $\left(\left(0,\ ialim^2 - (Iqcmd_{limzi}Iqcmd_x\left(FfhFflFvhFvlrecflag - recf\right.\right.\right.$	VarSer- vice
Ip-maxs	I_{pma}^2	FixPiecewise $\left(\left(0,\ ialim^2 - \frac{qref_0^2u^2}{v^2} \leq 0\right), \left(ialim^2 - \frac{qref_0^2u^2}{v^2},\ \text{True}\right)\right)$	Const-Service
Iq-maxs	I_{qma}^2	FixPiecewise $\left(\left(0,\ ialim^2 - (Ipcmd_{limzi}Ipcmd_x\left(FfhFflFvhFvlrecflag - recf\right.\right.\right.$	VarSer- vice

Discretes

Name	Symbol	Type	Info
SWPQ	SW_{PQ}	Switcher	
FL1	$FL1$	Limiter	Under frequency comparer
FL2	$FL2$	Limiter	Over frequency comparer
DB_db	db_{DB}	DeadBand	
VL1	$VL1$	Limiter	Under voltage comparer
VL2	$VL2$	Limiter	Over voltage comparer
VLo	VLo	Limiter	Voltage lower limit (0.01) flag
PHL	PHL	Limiter	limiter for Psum in [0, pmx]
VQ1	$VQ1$	Limiter	Under voltage comparer for Q droop
VQ2	$VQ2$	Limiter	Over voltage comparer for Q droop
Ipcmd_lim	lim_{Ipcmd}	HardLimiter	
Iqcmd_lim	lim_{Iqcmd}	HardLimiter	
LTN	LTN	LessThan	
SOClim	$SOClim$	HardLimiter	

Blocks

Name	Symbol	Type	Info
DB	DB	DeadBand1	frequency deviation deadband with gain
Ipcmd	I^{pcmd}	GainLimiter	Ip with limiter and coeff.
Iqcmd	I^{qcmd}	GainLimiter	Iq with limiter and coeff.
Ipout	$Ipout$	Lag	Output Ip filter
Iqout	$Iqout$	Lag	Output Iq filter
pIG	pIG	Integrator	State of charge

Setpoints

Name	Target	Info
pref	pref0	Initial P for the PVD1 device
qref	qref0	Initial Q for the PVD1 device
paux	Pext0	External additional signal added to Pext

Config Fields in [ESD1]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)
plim	P_{lim}	1	enable input power limit check bound by [0, pmx]	(0, 1)

EV1

Electric vehicle model type 1.

Modified from ESD1 model by adding the minumum power limit *pmn*. Like *pmx*, *pmn* acts on *Psum*, the sum of the active power references.

The limiter that uses *pmx* and *pmn* is enabled by default.

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
bus		interface bus id			mandatory
gen		static generator index			mandatory
Sn	S_n	device MVA rating	100	*MVA*	
fn	f_n	nominal frequency	60	*Hz*	
busf		Optional BusFreq measurement device idx			
xc	x_c	coupling reactance	0	*p.u.*	z
pqflag		P/Q priority for I limit; 0-Q priority, 1-P priority		*bool*	mandatory
igreg		Remote bus idx for droop response, None for local			
qmx	q_{mx}	Max. reactive power command	0.330	*pu*	power
qmn	q_{mn}	Min. reactive power command	-0.330	*pu*	power
pmx	p_{mx}	maximum power limit	9999	*pu*	power
v0	v_0	Lower limit of deadband for Vdroop response	0.800	*pu*	non_zero
v1	v_1	Upper limit of deadband for Vdroop response	1.100	*pu*	non_zero
dqdv	dq/dv	Q-V droop characteristics (negative)	-1		non_zero,power
fdbd	f_{dbd}	frequency deviation deadband	-0.017	*Hz*	non_positive
ddn	D_{dn}	Gain after f deadband	0	*pu (MW)/Hz*	non_negative,pov
ialim	I_{alim}	Apparent power limit	1.300		non_zero,non_ne
vt0	V_{t0}	Voltage tripping response curve point 0	0.880	*p.u.*	non_zero,non_ne
vt1	V_{t1}	Voltage tripping response curve point 1	0.900	*p.u.*	non_zero,non_ne
vt2	V_{t2}	Voltage tripping response curve point 2	1.100	*p.u.*	non_zero,non_ne

continues o

Table 4 – continued from previous page

Name	Symbol	Description	Default	Unit	Properties
vt3	V_{t3}	Voltage tripping response curve point 3	1.200	*p.u.*	non_zero,non_ne
vrflag	z_{VR}	V-trip is latching (0) or self-resetting (0-1)	0		
ft0	f_{t0}	Frequency tripping response curve point 0	59.500	*Hz*	non_zero,non_ne
ft1	f_{t1}	Frequency tripping response curve point 1	59.700	*Hz*	non_zero,non_ne
ft2	f_{t2}	Frequency tripping response curve point 2	60.300	*Hz*	non_zero,non_ne
ft3	f_{t3}	Frequency tripping response curve point 3	60.500	*Hz*	non_zero,non_ne
frflag	z_{FR}	f-trip is latching (0) or self-resetting (0-1)	0		
tip	T_{ip}	Inverter active current lag time constant	0.020	*s*	non_negative
tiq	T_{iq}	Inverter reactive current lag time constant	0.020	*s*	non_negative
gammap	γ_p	Ratio of PVD1.pref0 w.r.t to that of static PV	1		
gammaq	γ_q	Ratio of PVD1.qref0 w.r.t to that of static PV	1		
recflag	z_{rec}	Enable flag for voltage and frequency recovery limiters	1		
Tf	T_f	Integrator constant for SOC model	1		
SOCmin	SOC_{min}	Minimum required value for SOC in limiter	0		
SOCmax	SOC_{max}	Maximum allowed value for SOC in limiter	1		
SOCinit	SOC_{init}	Initial state of charge	0.500		
En	E_n	Rated energy capacity	100	*MWh*	
EtaC	Eta_C	Efficiency during charging	1		
EtaD	Eta_D	Efficiency during discharging	1		
pmn	p_{mn}	minimum power limit	-999	*pu*	power
ue	u_e	effective online status	0		

Variables

Name	Symbol	Type	Description	Unit	Properties
Ipout_y	$Ipout_y$	State	State in lag transfer function		v_str
Iqout_y	$Iqout_y$	State	State in lag transfer function		v_str
pIG_y	pIG_y	State	Integrator output		v_str
SOC	SOC	AliasState	Alias for state of charge		
fHz	fHz	Algeb	frequency in Hz	*Hz*	v_str
Ffl	Ffl	Algeb	Coeff. for under frequency		v_str
Ffh	Ffh	Algeb	Coeff. for over frequency		v_str
Fdev	$Fdev$	Algeb	Frequency deviation	*Hz*	v_str
DB_y	DB_y	Algeb	Deadband type 1 output		v_str
Fvl	Fvl	Algeb	Coeff. for under voltage		v_str
Fvh	Fvh	Algeb	Coeff. for over voltage		v_str
vp	vp	Algeb	Sensed positive voltage		v_str
Pext	$Pext$	Algeb	External power signal (for AGC)		v_str
Pref	$Pref$	Algeb	Reference power signal (for scheduling setpoint)		v_str
Psum	$Psum$	Algeb	Sum of P signals		v_str
Qdrp	$Qdrp$	Algeb	External power signal (for AGC)		v_str
Qref	$Qref$	Algeb	Reference power signal (for scheduling setpoint)		v_str
Qsum	$Qsum$	Algeb	Sum of Q signals		v_str
Ipul	$Ipul$	Algeb	Ipcmd before Ip hard limit		v_str
Iqul	$Iqul$	Algeb	Iqcmd before Iq hard limit		v_str
Ipmax	$Ipmax$	Algeb	Upper limit of Ip		v_str
Iqmax	$Iqmax$	Algeb	Upper limit of Iq		v_str
Ipcmd_x	$Ipcmd_x$	Algeb	Value before limiter		v_str
Iqcmd_x	$Iqcmd_x$	Algeb	Value before limiter		v_str
Ipmin	$Ipmin$	Algeb	Minimum value of Ip		v_str
a	a	ExtAlgeb	bus (or igreg) phase angle	*rad.*	
v	v	ExtAlgeb	bus (or igreg) terminal voltage	*p.u.*	
f	f	ExtAlgeb	Bus frequency	*p.u.*	

Initialization Equations

Nam	Sym bol	Type	Initial Value
Ipout	$Ipou$	State	$1.0Ipcmd_{limzi}Ipcmd_x\left(FfhFflFvhFvlrecflag - recflag + 1\right) + 1.0Ipcmd_{limzu}Ipmax\left(FfhFflFvhFvlrecflag - recflag + 1\right)$
Iqout	$Iqou$	State	$1.0Iqcmd_{limzi}Iqcmd_x\left(FfhFflFvhFvlrecflag - recflag + 1\right) - 1.0Iqcmd_{limzl}Iqmax\left(FfhFflFvhFvlrecflag - recflag + 1\right) + 1.0Iqcmd_{limzu}Iqmax\left(FfhFflFvhFvlrecflag - recflag + 1\right)$
pIG_	$pIG_{\imath}$	State	$SOCinit$
SOC	SOC	Alias	
fHz	fHz	Algeb	ffn
Ffl	Ffl	Algeb	$FL_{1zi}Kft_{01}\left(fHz - ft_0\right) + FL_{1zu}$
Ffh	Ffh	Algeb	$FL_{2zi}\left(Kft_{23}\left(-fHz + ft_2\right) + 1\right) + FL_{2zl}$
Fdev	$Fdev$	Algeb	$-fHz + fn$
DB_y	DB_y	Algeb	$ddn\left(DB_{dbzl}\left(Fdev - fdbd\right) + DB_{dbzu}\left(Fdev - 999.0\right)\right)$
Fvl	Fvl	Algeb	$Kvt_{01}VL_{1zi}\left(v - vt_0\right) + VL_{1zu}$
Fvh	Fvh	Algeb	$VL_{2zi}\left(Kvt_{23}\left(-v + vt_2\right) + 1\right) + VL_{2zl}$
vp	vp	Algeb	$VLo_{zi}v + 0.01VLo_{zl}$
Pext	$Pext$	Algeb	$Pext_0u$
Pref	$Pref$	Algeb	$pref_0u$
Psum	$Psum$	Algeb	$u\left(DB_y + Pext + Pref\right)$
Qdrp	$Qdrp$	Algeb	$VQ_{1zi}ue\left(dqdv\left(-Vcomp + Vqu\right) + qmx\right) + VQ_{1zl}qmxue + VQ_{2zi}dqdvue\left(-Vcomp + v_1\right) + VQ_{2zu}qmn$
Qref	$Qref$	Algeb	$qref_0u$
Qsum	$Qsum$	Algeb	$u\left(VQ_{1zi}ue\left(dqdv\left(-Vcomp + Vqu\right) + qmx\right) + VQ_{1zl}qmxue + VQ_{2zi}dqdvue\left(-Vcomp + v_1\right) +$
Ipul	$Ipul$	Algeb	$\frac{PHL_{zi}Psum + PHL_{zl}pmn + PHL_{zu}pmx}{vp}$
Iqul	$Iqul$	Algeb	$\frac{Qsum}{vp}$
Ipmax	$Ipmax$	Algeb	$\left(1 - SOClim_{zl}\right)\left(\sqrt{Ipmaxsq_0}SWPQ_{s0} + SWPQ_{s1}ialim\right)$
Iqmax	$Iqmax$	Algeb	$\sqrt{Iqmaxsq_0}SWPQ_{s1} + SWPQ_{s0}ialim$
Ipcm	$Ipcm$	Algeb	$Ipul$
Iqcm	$Iqcm$	Algeb	$Iqul$

Differential Equations

Nam	Sym bol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
Ipout	$Ipou$	State	$1.0Ipcmd_{limzi}Ipcmd_x\left(FfhFflFvhFvlrecflag - recflag + 1\right) + 1.0Ipcmd_{limzu}Ipmax\left(FfhFflFvhFvlrecflag - recflag + 1\right) - Ipout_y$	T_{ip}
Iqout	$Iqou$	State	$1.0Iqcmd_{limzi}Iqcmd_x\left(FfhFflFvhFvlrecflag - recflag + 1\right) - 1.0Iqcmd_{limzl}Iqmax\left(FfhFflFvhFvlrecflag - recflag + 1\right) + 1.0Iqcmd_{limzu}Iqmax\left(FfhFflFvhFvlrecflag - recflag + 1\right) - Iqout_y$	T_{iq}
pIG_	$pIG_{\imath}$	State	$\frac{S_{b,sys}\left(-EtaCIpout_yLTN_{z1}v - \frac{Ipout_yLTN_{z0}v}{EtaD}\right)}{3600En}$	T_f
SOC	SOC	Alias	0	

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
fHz	fHz	Algeb	$ffn - fHz$
Ffl	Ffl	Algeb	$FL_{1zi}Kft_{01}\left(fHz - ft_0\right) + FL_{1zu} - Ffl$
Ffh	Ffh	Algeb	$FL_{2zi}\left(Kft_{23}\left(-fHz + ft_2\right) + 1\right) + FL_{2zl} - Ffh$
Fdev	$Fdev$	Algeb	$-Fdev - fHz + fn$
DB_y	DB_y	Algeb	$-DB_y + ddn\left(DB_{dbzl}\left(Fdev - fdbd\right) + DB_{dbzu}\left(Fdev - 999.0\right)\right)$
Fvl	Fvl	Algeb	$-Fvl + Kvt_{01}VL_{1zi}\left(v - vt_0\right) + VL_{1zu}$
Fvh	Fvh	Algeb	$-Fvh + VL_{2zi}\left(Kvt_{23}\left(-v + vt_2\right) + 1\right) + VL_{2zl}$
vp	vp	Algeb	$VLo_{zi}v + 0.01VLo_{zl} - vp$
Pext	$Pext$	Algeb	$-Pext + Pext_0ue$
Pref	$Pref$	Algeb	$-Pref + pref_0ue$
Psum	$Psum$	Algeb	$-Psum + ue\left(DB_y + Pext + Pref\right)$
Qdrp	$Qdrp$	Algeb	$-Qdrp + VQ_{1zi}ue\left(dqdv\left(-Vcomp + Vqu\right) + qmx\right) + VQ_{1zl}qmxue + VQ_{2zi}dqdvue\left(-Vcomp + v_1\right) + VQ_{2zu}qmn$
Qref	$Qref$	Algeb	$-Qref + qref_0ue$
Qsum	$Qsum$	Algeb	$-Qsum + ue\left(Qdrp + Qref\right)$
Ipul	$Ipul$	Algeb	$-Ipul + \frac{PHL_{zi}Psum + PHL_{zl}pmn + PHL_{zu}pmx}{vp}$
Iqul	$Iqul$	Algeb	$-Iqul + \frac{Qsum}{vp}$
Ipmax	$Ipmax$	Algeb	$-Ipmax + \left(1 - SOClim_{zl}\right)\left(\sqrt{Ipmaxsq}SWPQ_{s0} + SWPQ_{s1}ialim\right)$
Iqmax	$Iqmax$	Algeb	$-Iqmax + \sqrt{Iqmaxsq}SWPQ_{s1} + SWPQ_{s0}ialim$
Ipcmd_	$Ipcmd$	Algeb	$-Ipcmd_x + Ipul$
Iqcmd_	$Iqcmd$	Algeb	$-Iqcmd_x + Iqul$
Ipmin	$Ipmin$	Algeb	$-Ipmin + \left(SOClim_{zu} - 1\right)\left(\sqrt{Ipmaxsq}SWPQ_{s0} + SWPQ_{s1}ialim\right)$
a	a	ExtAlgeb	$-Ipout_yuev$
v	v	ExtAlgeb	$-Iqout_yuev$
f	f	ExtAlgeb	0

Services

Nam	Sym bol	Equation	Type
pref0	P_{ref0}	$gammapp0s$	Const-Service
qref0	Q_{ref0}	$gammaqq0s$	Const-Service
Kft0	K_{ft0}	$\frac{1}{-ft_0+ft_1}$	Const-Service
Kft2	K_{ft2}	$\frac{1}{-ft_2+ft_3}$	Const-Service
Kvt0	K_{vt0}	$\frac{1}{-vt_0+vt_1}$	Const-Service
Kvt2	K_{vt2}	$\frac{1}{-vt_2+vt_3}$	Const-Service
Pext0	P_{ext0}	0	Const-Service
Vcon	V_{com}	$\sqrt{Ipout_y^2xc^2 - iIpout_yvxce^{ia} + iIpout_yvxce^{-ia} + Iqout_y^2xc^2 - Iqout_yvxce^{ia} - I}$	VarService
Vqu	V_{qu}	$v_1 - \frac{-qmn+qref_0}{dqdv}$	Const-Service
Vql	V_{ql}	$v_0 + \frac{qmx-qref_0}{dqdv}$	Const-Service
Ip-maxs	I_{pma}^2	FixPiecewise $\left(\left(0,\ ialim^2 - (Iqcmd_{limzi}Iqcmd_x\left(FfhFflFvhFvlrecflag - recf\right.\right.\right.$	VarService
Ip-maxs	I_{pma}^2	FixPiecewise $\left(\left(0,\ ialim^2 - \frac{qref_0^2u^2}{v^2} \leq 0\right), \left(ialim^2 - \frac{qref_0^2u^2}{v^2},\ \text{True}\right)\right)$	Const-Service
Iq-maxs	I_{qma}^2	FixPiecewise $\left(\left(0,\ ialim^2 - (Ipcmd_{limzi}Ipcmd_x\left(FfhFflFvhFvlrecflag - recf\right.\right.\right.$	VarService

Discretes

Name	Symbol	Type	Info
SWPQ	SW_{PQ}	Switcher	
FL1	$FL1$	Limiter	Under frequency comparer
FL2	$FL2$	Limiter	Over frequency comparer
DB_db	db_{DB}	DeadBand	
VL1	$VL1$	Limiter	Under voltage comparer
VL2	$VL2$	Limiter	Over voltage comparer
VLo	VLo	Limiter	Voltage lower limit (0.01) flag
PHL	PHL	Limiter	limiter for Psum in [pmn, pmx]
VQ1	$VQ1$	Limiter	Under voltage comparer for Q droop
VQ2	$VQ2$	Limiter	Over voltage comparer for Q droop
Ipcmd_lim	lim_{Ipcmd}	HardLimiter	
Iqcmd_lim	lim_{Iqcmd}	HardLimiter	
LTN	LTN	LessThan	
SOClim	$SOClim$	HardLimiter	

Blocks

Name	Symbol	Type	Info
DB	DB	DeadBand1	frequency deviation deadband with gain
Ipcmd	I^{pcmd}	GainLimiter	Ip with limiter and coeff.
Iqcmd	I^{qcmd}	GainLimiter	Iq with limiter and coeff.
Ipout	$Ipout$	Lag	Output Ip filter
Iqout	$Iqout$	Lag	Output Iq filter
pIG	pIG	Integrator	State of charge

Setpoints

Name	Target	Info
pref	pref0	Initial P for the PVD1 device
qref	qref0	Initial Q for the PVD1 device
paux	Pext0	External additional signal added to Pext

Config Fields in [EV1]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)
plim	P_{lim}	1	enable input power limit check bound by [0, pmx]	(0, 1)

EV2

Electric vehicle model type 2.

Derived from EV1, EV2 introduces *pcap* multiplied to *pmx*.

Psum will be limited to [pmn, pmx * pcap].

The model does not check the signs or values of *pmn*, *pmx*, or *pcap*. The input data is required to satisfy *pmn* <= *pmx* * *pcap*.

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
bus		interface bus id			mandatory
gen		static generator index			mandatory
Sn	S_n	device MVA rating	100	*MVA*	
fn	f_n	nominal frequency	60	*Hz*	
busf		Optional BusFreq measurement device idx			
xc	x_c	coupling reactance	0	*p.u.*	z
pqflag		P/Q priority for I limit; 0-Q priority, 1-P priority		*bool*	mandatory
igreg		Remote bus idx for droop response, None for local			
qmx	q_{mx}	Max. reactive power command	0.330	*pu*	power
qmn	q_{mn}	Min. reactive power command	-0.330	*pu*	power
pmx	p_{mx}	maximum power limit	9999	*pu*	power
v0	v_0	Lower limit of deadband for Vdroop response	0.800	*pu*	non_zero
v1	v_1	Upper limit of deadband for Vdroop response	1.100	*pu*	non_zero
dqdv	dq/dv	Q-V droop characteristics (negative)	-1		non_zero,power
fdbd	f_{dbd}	frequency deviation deadband	-0.017	*Hz*	non_positive
ddn	D_{dn}	Gain after f deadband	1	*pu (MW)/Hz*	non_negative,pov
ialim	I_{alim}	Apparent power limit	1.300		non_zero,non_ne
vt0	V_{t0}	Voltage tripping response curve point 0	0.880	*p.u.*	non_zero,non_ne

continues o

Table 5 – continued from previous page

Name	Symbol	Description	Default	Unit	Properties
vt1	V_{t1}	Voltage tripping response curve point 1	0.900	*p.u.*	non_zero,non_ne
vt2	V_{t2}	Voltage tripping response curve point 2	1.100	*p.u.*	non_zero,non_ne
vt3	V_{t3}	Voltage tripping response curve point 3	1.200	*p.u.*	non_zero,non_ne
vrflag	z_{VR}	V-trip is latching (0) or self-resetting (0-1)	0		
ft0	f_{t0}	Frequency tripping response curve point 0	59.500	*Hz*	non_zero,non_ne
ft1	f_{t1}	Frequency tripping response curve point 1	59.700	*Hz*	non_zero,non_ne
ft2	f_{t2}	Frequency tripping response curve point 2	60.300	*Hz*	non_zero,non_ne
ft3	f_{t3}	Frequency tripping response curve point 3	60.500	*Hz*	non_zero,non_ne
frflag	z_{FR}	f-trip is latching (0) or self-resetting (0-1)	0		
tip	T_{ip}	Inverter active current lag time constant	0.020	*s*	non_negative
tiq	T_{iq}	Inverter reactive current lag time constant	0.020	*s*	non_negative
gammap	γ_p	Ratio of PVD1.pref0 w.r.t to that of static PV	1		
gammaq	γ_q	Ratio of PVD1.qref0 w.r.t to that of static PV	1		
recflag	z_{rec}	Enable flag for voltage and frequency recovery limiters	1		
Tf	T_f	Integrator constant for SOC model	1		
SOCmin	SOC_{min}	Minimum required value for SOC in limiter	0		
SOCmax	SOC_{max}	Maximum allowed value for SOC in limiter	1		
SOCinit	SOC_{init}	Initial state of charge	0.500		
En	E_n	Rated energy capacity	100	*MWh*	
EtaC	Eta_C	Efficiency during charging	1		
EtaD	Eta_D	Efficiency during discharging	1		
pmn	p_{mn}	minimum power limit	-999	*pu*	power
pcap	p_{cap}	power ratio multiplied to pmx in [-1, 1]	0		
ue	u_e	effective online status	0		

Variables

Name	Symbol	Type	Description	Unit	Properties
Ipout_y	$Ipout_y$	State	State in lag transfer function		v_str
Iqout_y	$Iqout_y$	State	State in lag transfer function		v_str
pIG_y	pIG_y	State	Integrator output		v_str
SOC	SOC	AliasState	Alias for state of charge		
fHz	fHz	Algeb	frequency in Hz	*Hz*	v_str
Ffl	Ffl	Algeb	Coeff. for under frequency		v_str
Ffh	Ffh	Algeb	Coeff. for over frequency		v_str
Fdev	$Fdev$	Algeb	Frequency deviation	*Hz*	v_str
DB_y	DB_y	Algeb	Deadband type 1 output		v_str
Fvl	Fvl	Algeb	Coeff. for under voltage		v_str
Fvh	Fvh	Algeb	Coeff. for over voltage		v_str
vp	vp	Algeb	Sensed positive voltage		v_str
Pext	$Pext$	Algeb	External power signal (for AGC)		v_str
Pref	$Pref$	Algeb	Reference power signal (for scheduling setpoint)		v_str
Psum	$Psum$	Algeb	Sum of P signals		v_str
Qdrp	$Qdrp$	Algeb	External power signal (for AGC)		v_str
Qref	$Qref$	Algeb	Reference power signal (for scheduling setpoint)		v_str
Qsum	$Qsum$	Algeb	Sum of Q signals		v_str
Ipul	$Ipul$	Algeb	Ipcmd before Ip hard limit		v_str
Iqul	$Iqul$	Algeb	Iqcmd before Iq hard limit		v_str
Ipmax	$Ipmax$	Algeb	Upper limit of Ip		v_str
Iqmax	$Iqmax$	Algeb	Upper limit of Iq		v_str
Ipcmd_x	$Ipcmd_x$	Algeb	Value before limiter		v_str
Iqcmd_x	$Iqcmd_x$	Algeb	Value before limiter		v_str
Ipmin	$Ipmin$	Algeb	Minimum value of Ip		v_str
PHLup	$PHLup$	Algeb	PHL upper limit		v_str
a	a	ExtAlgeb	bus (or igreg) phase angle	*rad.*	
v	v	ExtAlgeb	bus (or igreg) terminal voltage	*p.u.*	
f	f	ExtAlgeb	Bus frequency	*p.u.*	

Initialization Equations

Nam	Sym bol	Type	Initial Value
Ipout	$Ipou$	State	$1.0Ipcmd_{limzi}Ipcmd_x\left(FfhFflFvhFvlrecflag - recflag + 1\right) + 1.0Ipcmd_{limzu}Ipmax\left(FfhFflFvhFvlrecflag - recflag + 1\right)$
Iqout	$Iqou$	State	$1.0Iqcmd_{limzi}Iqcmd_x\left(FfhFflFvhFvlrecflag - recflag + 1\right) - 1.0Iqcmd_{limzl}Iqmax\left(FfhFflFvhFvlrecflag - recflag + 1\right) + 1.0Iqcmd_{limzu}Iqmax\left(FfhFflFvhFvlrecflag - recflag + 1\right)$
pIG_	$pIG_{\imath}$	State	$SOCinit$
SOC	SOC	Alias!	
fHz	fHz	Al-geb	ffn
Ffl	Ffl	Al-geb	$FL_{1zi}Kft_{01}\left(fHz - ft_0\right) + FL_{1zu}$
Ffh	Ffh	Al-geb	$FL_{2zi}\left(Kft_{23}\left(-fHz + ft_2\right) + 1\right) + FL_{2zl}$
Fdev	$Fdev$	Al-geb	$-fHz + fn$
DB_y	DB_y	Al-geb	$ddn\left(DB_{dbzl}\left(Fdev - fdbd\right) + DB_{dbzu}\left(Fdev - 999.0\right)\right)$
Fvl	Fvl	Al-geb	$Kvt_{01}VL_{1zi}\left(v - vt_0\right) + VL_{1zu}$
Fvh	Fvh	Al-geb	$VL_{2zi}\left(Kvt_{23}\left(-v + vt_2\right) + 1\right) + VL_{2zl}$
vp	vp	Al-geb	$VLo_{zi}v + 0.01VLo_{zl}$
Pext	$Pext$	Al-geb	$Pext_0u$
Pref	$Pref$	Al-geb	$pref_0u$
Psum	$Psum$	Al-geb	$u\left(DB_y + Pext + Pref\right)$
Qdrp	$Qdrp$	Al-geb	$VQ_{1zi}ue\left(dqdv\left(-Vcomp + Vqu\right) + qmx\right) + VQ_{1zl}qmxue + VQ_{2zi}dqdvue\left(-Vcomp + v_1\right) + VQ_{2zu}qmn$
Qref	$Qref$	Al-geb	$qref_0u$
Qsum	$Qsum$	Al-geb	$u\left(VQ_{1zi}ue\left(dqdv\left(-Vcomp + Vqu\right) + qmx\right) + VQ_{1zl}qmxue + VQ_{2zi}dqdvue\left(-Vcomp + v_1\right) +$
Ipul	$Ipul$	Al-geb	$\frac{PHL_{2zi}Psum + PHL_{2zl}pmn + PHL_{2zu}PHLup}{vp}$
Iqul	$Iqul$	Al-geb	$\frac{Qsum}{vp}$
Ip-max	$Ipmax$	Al-geb	$\left(1 - SOClim_{zl}\right)\left(\sqrt{Ipmaxsq_0}SWPQ_{s0} + SWPQ_{s1}ialim\right)$
Iq-max	$Iqmax$	Al-geb	$\sqrt{Iqmaxsq_0}SWPQ_{s1} + SWPQ_{s0}ialim$
Ipcm	$Ipcm$	Al-geb	$Ipul$
Iqcm	$Iqcm$	Al-geb	$Iqul$

Differential Equations

Nam	Sym bol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
Ipout	$Ipou$	State	$1.0Ipcmd_{limzi}Ipcmd_x\left(FfhFflFvhFvlrecflag - recflag + 1\right) + 1.0Ipcmd_{limzu}Ipmax\left(FfhFflFvhFvlrecflag - recflag + 1\right) - Ipout_y$	T_{ip}
Iqout	$Iqou$	State	$1.0Iqcmd_{limzi}Iqcmd_x\left(FfhFflFvhFvlrecflag - recflag + 1\right) - 1.0Iqcmd_{limzl}Iqmax\left(FfhFflFvhFvlrecflag - recflag + 1\right) + 1.0Iqcmd_{limzu}Iqmax\left(FfhFflFvhFvlrecflag - recflag + 1\right) - Iqout_y$	T_{iq}
pIG_	$pIG_{\imath}$	State	$\frac{S_{b,sys}\left(-EtaCIpout_yLTN_{z1}v - \frac{Ipout_yLTN_{z0}v}{EtaD}\right)}{3600En}$	T_f
SOC	SOC	Alias	0	

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
fHz	fHz	Algeb	$ffn - fHz$
Ffl	Ffl	Algeb	$FL_{1zi}Kft_{01}\left(fHz - ft_0\right) + FL_{1zu} - Ffl$
Ffh	Ffh	Algeb	$FL_{2zi}\left(Kft_{23}\left(-fHz + ft_2\right) + 1\right) + FL_{2zl} - Ffh$
Fdev	$Fdev$	Algeb	$-Fdev - fHz + fn$
DB_y	DB_y	Algeb	$-DB_y + ddn\left(DB_{dbzl}\left(Fdev - fdbd\right) + DB_{dbzu}\left(Fdev - 999.0\right)\right)$
Fvl	Fvl	Algeb	$-Fvl + Kvt_{01}VL_{1zi}\left(v - vt_0\right) + VL_{1zu}$
Fvh	Fvh	Algeb	$-Fvh + VL_{2zi}\left(Kvt_{23}\left(-v + vt_2\right) + 1\right) + VL_{2zl}$
vp	vp	Algeb	$VLo_{zi}v + 0.01VLo_{zl} - vp$
Pext	$Pext$	Algeb	$-Pext + Pext_0ue$
Pref	$Pref$	Algeb	$-Pref + pref_0ue$
Psum	$Psum$	Algeb	$-Psum + ue\left(DB_y + Pext + Pref\right)$
Qdrp	$Qdrp$	Algeb	$-Qdrp + VQ_{1zi}ue\left(dqdv\left(-Vcomp + Vqu\right) + qmx\right) + VQ_{1zl}qmxue + VQ_{2zi}dqdvue\left(-Vcomp + v_1\right) + VQ_{2zu}qmn$
Qref	$Qref$	Algeb	$-Qref + qref_0ue$
Qsum	$Qsum$	Algeb	$-Qsum + ue\left(Qdrp + Qref\right)$
Ipul	$Ipul$	Algeb	$-Ipul + \frac{PHL_{2zi}Psum + PHL_{2zl}pmn + PHL_{2zu}PHLup}{vp}$
Iqul	$Iqul$	Algeb	$-Iqul + \frac{Qsum}{vp}$
Ipmax	$Ipmax$	Algeb	$-Ipmax + \left(1 - SOClim_{zl}\right)\left(\sqrt{Ipmaxsq}SWPQ_{s0} + SWPQ_{s1}ialim\right)$
Iqmax	$Iqmax$	Algeb	$-Iqmax + \sqrt{Iqmaxsq}SWPQ_{s1} + SWPQ_{s0}ialim$
Ipcmd_	$Ipcmd$	Algeb	$-Ipcmd_x + Ipul$
Iqcmd_	$Iqcmd$	Algeb	$-Iqcmd_x + Iqul$
Ipmin	$Ipmin$	Algeb	$-Ipmin + \left(SOClim_{zu} - 1\right)\left(\sqrt{Ipmaxsq}SWPQ_{s0} + SWPQ_{s1}ialim\right)$
PHLup	$PHLup$	Algeb	$-PHLup + pcappmx$
a	a	ExtAlgeb	$-Ipout_yuev$
v	v	ExtAlgeb	$-Iqout_yuev$
f	f	ExtAlgeb	0

Services

Nam	Sym bol	Equation	Type
pref0	P_{ref0}	$gammapp0s$	Const-Service
qref0	Q_{ref0}	$gammaqq0s$	Const-Service
Kft0	K_{ft0}	$\frac{1}{-ft_0+ft_1}$	Const-Service
Kft2	K_{ft2}	$\frac{1}{-ft_2+ft_3}$	Const-Service
Kvt0	K_{vt0}	$\frac{1}{-vt_0+vt_1}$	Const-Service
Kvt2	K_{vt2}	$\frac{1}{-vt_2+vt_3}$	Const-Service
Pext0	P_{ext0}	0	Const-Service
Vcon	V_{com}	$\sqrt{Ipout_y^2xc^2 - iIpout_yvxce^{ia} + iIpout_yvxce^{-ia} + Iqout_y^2xc^2 - Iqout_yvxce^{ia} - I}$	VarService
Vqu	V_{qu}	$v_1 - \frac{-qmn+qref_0}{dqdv}$	Const-Service
Vql	V_{ql}	$v_0 + \frac{qmx-qref_0}{dqdv}$	Const-Service
Ip-maxs	I^2_{pma}	FixPiecewise $\left(\left(0,\ ialim^2 - (Iqcmd_{limzi}Iqcmd_x\left(FfhFflFvhFvlrecflag - recf\right.\right.\right.$	VarService
Ip-maxs	I^2_{pma}	FixPiecewise $\left(\left(0,\ ialim^2 - \frac{qref_0^2u^2}{v^2} \leq 0\right), \left(ialim^2 - \frac{qref_0^2u^2}{v^2},\ \text{True}\right)\right)$	Const-Service
Iq-maxs	I^2_{qma}	FixPiecewise $\left(\left(0,\ ialim^2 - (Ipcmd_{limzi}Ipcmd_x\left(FfhFflFvhFvlrecflag - recf\right.\right.\right.$	VarService

Discretes

Name	Symbol	Type	Info
SWPQ	SW_{PQ}	Switcher	
FL1	$FL1$	Limiter	Under frequency comparer
FL2	$FL2$	Limiter	Over frequency comparer
DB_db	db_{DB}	DeadBand	
VL1	$VL1$	Limiter	Under voltage comparer
VL2	$VL2$	Limiter	Over voltage comparer
VLo	VLo	Limiter	Voltage lower limit (0.01) flag
PHL	PHL	Limiter	limiter for Psum in [pmn, pmx]
VQ1	$VQ1$	Limiter	Under voltage comparer for Q droop
VQ2	$VQ2$	Limiter	Over voltage comparer for Q droop
Ipcmd_lim	lim_{Ipcmd}	HardLimiter	
Iqcmd_lim	lim_{Iqcmd}	HardLimiter	
LTN	LTN	LessThan	
SOClim	$SOClim$	HardLimiter	
PHL2	$PHL2$	Limiter	limiter for Psum in [pmn, pcap * pmx]

Blocks

Name	Symbol	Type	Info
DB	DB	DeadBand1	frequency deviation deadband with gain
Ipcmd	I^{pcmd}	GainLimiter	Ip with limiter and coeff.
Iqcmd	I^{qcmd}	GainLimiter	Iq with limiter and coeff.
Ipout	$Ipout$	Lag	Output Ip filter
Iqout	$Iqout$	Lag	Output Iq filter
pIG	pIG	Integrator	State of charge

Setpoints

Name	Target	Info
pref	pref0	Initial P for the PVD1 device
qref	qref0	Initial Q for the PVD1 device
paux	Pext0	External additional signal added to Pext

Config Fields in [EV2]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)
plim	P_{lim}	1	enable input power limit check bound by [0, pmx]	(0, 1)

5.3.9 DGProtection

Protection model for DG.

Common Parameters: u, name

Available models: *DGPRCT1*, *DGPRCTExt*

DGPRCT1

DGPRCT1 model, follow IEEE-1547-2018. DGPRCT stands for DG protection.

A demo is provided: examples/demonstration/1.1 demo_DGPRCT1.ipynb

Target device (limited to DG group) `Psum` and `Qsum` will decrease to zero immediately when frequency/voltage protection flag is raised. Once the lock is released, `Psum` and `Qsum` will return to normal immediately.

DG group base model `PVD1` already has a degrading function which is used to degrade output under abnormal condition. it is recommended to turn it off by setting *recflag = 0*.

`fen` and `Ven` are protection enabling parameters. 1/0 is on/off.

`lock` is lock flag signal.

It should be noted that, the lock only lock the `fHz` (frequency read value) of DG model. The source values (which come from `BusFreq` f remain unchanged.)

Protection sensors (e.g., IAWfl1) are instances of `IntergratorAntiWindup`. All the protection sensors will be reset after `lock` returns to 0. Resetting action takes *Tres* to finish.

The model does not check the shedding points sequence. The input parameters are required to satisfy `fl3 < fl2 < fl1 < fu1 < fu2 < fu3`, and `ul4 < ul3 < ul2 < ul1 < uu1 < uu2 < uu3`.

Default settings:

Frequency (Hz):

(fl3, fl2), Tfl2 [(50.0, 57.5), 10s]

(fl2, fl1), Tfl1 [(57.5, 59.2), 300s]

(fu1, fu2), Tfu1 [(60.5, 61.5), 300s]

(fu2, fu3), Tfu2 [(61.5, 70.0), 10s]

Voltage (p.u.):

(vl4, vl3), Tvl3 [(0.10, 0.45), 0.16s]

(vl3, vl2), Tvl2 [(0.45, 0.60), 1s]

(vl2, vl1), Tvl1 [(0.60, 0.88), 2s]

(vu1, vu2), Tvu1 [(1.10, 1.20), 1s]

(vu2, vu3), Tvu2 [(1.20, 2.00), 0.16s]

Reference:

NERC. Bulk Power System Reliability Perspectives on the Adoption of IEEE 1547-2018. March 2020. Available:

https://www.nerc.com/comm/PC_Reliability_Guidelines_DL/Guideline_IEEE_1547-2018_BPS_Perspectives.pdf

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
dev		idx of the target device			mandatory
busfreq		Target device interface bus measurement device idx			
fen	fen	Frequency deviation protection enable. 1 for enable, 0 for disable.	1		
Ven	Ven	Voltage deviation protection enable. 1 for enable, 0 for disable.	0		
fl3	$fl3$	Under frequency shadding point 3	50	*Hz*	
fl2	$fl2$	Over frequency shadding point 2	57.500	*Hz*	
fl1	$fl1$	Under frequency shadding point 1	59.200	*Hz*	
fu1	$fu1$	Over frequency shadding point 1	60.500	*Hz*	
fu2	$fu2$	Over frequency shadding point 2	61.500	*Hz*	
fu3	$fu3$	Over frequency shadding point 3	70	*Hz*	
Tfl1	T_{fl1}	Stand time for (fl2, fl1)	300		non_negative
Tfl2	T_{fl2}	Stand time for (fl3, fl2)	10		non_negative
Tfu1	T_{fu1}	Stand time for (fu1, fu2)	300		non_negative
Tfu2	T_{fu2}	Stand time for (fu2, fu3)	10		non_negative
vl4	$vl4$	Under voltage shadding point 4	0.100	*p.u.*	
vl3	$vl3$	Under voltage shadding point 3	0.450	*p.u.*	
vl2	$vl2$	Under voltage shadding point 2	0.600	*p.u.*	
vl1	$vl1$	Under voltage shadding point 1	0.880	*p.u.*	
vu1	$vu1$	Over voltage shadding point 1	1.100	*p.u.*	
vu2	$vu2$	Over voltage shadding point 2	1.200	*p.u.*	
vu3	$vu3$	Over voltage shadding point 3	2	*p.u.*	
Tvl1	T_{vl1}	Stand time for (vl2, vl1)	2		non_negative

continues on next page

Table 6 – continued from previous page

Name	Symbol	Description	Default	Unit	Properties
Tvl2	T_{vl2}	Stand time for (vl3, vl2)	1		non_negative
Tvl3	T_{vl3}	Stand time for (vl4, vl3)	0.160		non_negative
Tvu1	T_{vu1}	Stand time for (vu1, vu2)	1		non_negative
Tvu2	T_{vu2}	Stand time for (vu2, vu3)	0.160		non_negative
Tres		Integrator reset time	0.050		
ue	u_e	effective online status	0		
bus			0		
fn			0		

Variables

Name	Symbol	Type	Description	Unit	Properties
IAWfl1_y	$IAWfl_{1y}$	State	AW Integrator output		v_str
IAWfl2_y	$IAWfl_{2y}$	State	AW Integrator output		v_str
IAWfu1_y	$IAWfu_{1y}$	State	AW Integrator output		v_str
IAWfu2_y	$IAWfu_{2y}$	State	AW Integrator output		v_str
IAWVl1_y	$IAWVl_{1y}$	State	AW Integrator output		v_str
IAWVl2_y	$IAWVl_{2y}$	State	AW Integrator output		v_str
IAWVl3_y	$IAWVl_{3y}$	State	AW Integrator output		v_str
IAWVu1_y	$IAWVu_{1y}$	State	AW Integrator output		v_str
IAWVu2_y	$IAWVu_{2y}$	State	AW Integrator output		v_str
fHz	fHz	Algeb	frequency in Hz		v_str
dsum	$dsum$	Algeb	lock signal summation		v_str
lock	$lock$	Algeb	lock flag		v_str
f	f	ExtAlgeb	DG frequency read value	*p.u.*	
fin	fin	ExtAlgeb	original f from DG		
fHzl	$fHzl$	ExtAlgeb	Frequency measure lock		
Pext	$Pext$	ExtAlgeb	original Pext from DG		
Pref	$Pref$	ExtAlgeb	original Pref from DG		
Pdrp	$Pdrp$	ExtAlgeb	original Pdrp from DG		
Psum	$Psum$	ExtAlgeb	Active power lock		
Qdrp	$Qdrp$	ExtAlgeb	original Qdrp from DG		
Qref	$Qref$	ExtAlgeb	original Qref from DG		
Qsum	$Qsum$	ExtAlgeb	Reactive power lock		
v	v	ExtAlgeb	Bus voltage	*p.u.*	

Initialization Equations

Name	Symbol	Type	Initial Value
IAWfl1_y	$IAWfl_{1y}$	State	0
IAWfl2_y	$IAWfl_{2y}$	State	0
IAWfu1_y	$IAWfu_{1y}$	State	0
IAWfu2_y	$IAWfu_{2y}$	State	0
IAWVl1_y	$IAWVl_{1y}$	State	0
IAWVl2_y	$IAWVl_{2y}$	State	0
IAWVl3_y	$IAWVl_{3y}$	State	0
IAWVu1_y	$IAWVu_{1y}$	State	0
IAWVu2_y	$IAWVu_{2y}$	State	0
fHz	fHz	Algeb	ffn
dsum	$dsum$	Algeb	0
lock	$lock$	Algeb	0
f	f	ExtAlgeb	
fin	fin	ExtAlgeb	
fHzl	$fHzl$	ExtAlgeb	
Pext	$Pext$	ExtAlgeb	
Pref	$Pref$	ExtAlgeb	
Pdrp	$Pdrp$	ExtAlgeb	
Psum	$Psum$	ExtAlgeb	
Qdrp	$Qdrp$	ExtAlgeb	
Qref	$Qref$	ExtAlgeb	
Qsum	$Qsum$	ExtAlgeb	
v	v	ExtAlgeb	

Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
IAWfl1_y	$IAWfl_{1y}$	State	$Lfl_{1zi}\left(1-res\right)-\frac{Tfl_{1}res}{Tres}$	1
IAWfl2_y	$IAWfl_{2y}$	State	$Lfl_{2zi}\left(1-res\right)-\frac{Tfl_{2}res}{Tres}$	1
IAWfu1_y	$IAWfu_{1y}$	State	$Lfu_{1zi}\left(1-res\right)-\frac{Tfu_{1}res}{Tres}$	1
IAWfu2_y	$IAWfu_{2y}$	State	$Lfl_{2zi}\left(1-res\right)-\frac{Tfu_{2}res}{Tres}$	1
IAWVl1_y	$IAWVl_{1y}$	State	$LVl_{1zi}\left(1-res\right)-\frac{Tvl_{1}res}{Tres}$	1
IAWVl2_y	$IAWVl_{2y}$	State	$LVl_{2zi}\left(1-res\right)-\frac{Tvl_{2}res}{Tres}$	1
IAWVl3_y	$IAWVl_{3y}$	State	$LVl_{2zi}\left(1-res\right)-\frac{Tvl_{3}res}{Tres}$	1
IAWVu1_y	$IAWVu_{1y}$	State	$LVu_{1zi}\left(1-res\right)-\frac{Tvu_{1}res}{Tres}$	1
IAWVu2_y	$IAWVu_{2y}$	State	$LVu_{2zi}\left(1-res\right)-\frac{Tvu_{2}res}{Tres}$	1

Algebraic Equations

Nam	Sym bol	Type	RHS of Equation "0 = g(x, y)"
fHz	fHz	Al-geb	$ffn - fHz$
dsum	$dsun$	Al-geb	$Ven\left(IAWVl_{1limzu}LVl_{1zi} + IAWVl_{2limzu}LVl_{2zi} + IAWVl_{3limzu}LVl_{3zi} + IAWVu_{1limzu}LVu_{1}$ $dsum + fen\left(IAWfl_{1limzu}Lfl_{1zi} + IAWfl_{2limzu}Lfl_{2zi} + IAWfu_{1limzu}Lfu_{1zi} + IAWfu_{2limzu}L$
lock	$lock$	Al-geb	$Ldsum_{zu} - lock$
f	f	Ex-tAl-geb	0
fin	fin	Ex-tAl-geb	0
fHzl	fHz	Ex-tAl-geb	$-ffnlock$
Pext	$Pext$	Ex-tAl-geb	0
Pref	$Prej$	Ex-tAl-geb	0
Pdrp	$Pdrp$	Ex-tAl-geb	0
Psum	$Psun$	Ex-tAl-geb	$-lock\left(Pdrp + Pext + Pref\right)$
Qdrp	$Qdrp$	Ex-tAl-geb	0
Qref	$Qrej$	Ex-tAl-geb	0
Qsun	$Qsun$	Ex-tAl-geb	$-lock\left(Qdrp + Qref\right)$
v	v	Ex-tAl-geb	0

Services

Name	Symbol	Equation	Type
ltu	ltu	0.8	ConstService
ltl	ltl	0.2	ConstService
zero	$zero$	0	ConstService
res	res	0	ExtendedEvent

Discretes

Name	Symbol	Type	Info
Ldsum	$Ldsum$	Limiter	lock signal comparer, zu is to act
Lfl1	$Lfl1$	Limiter	Frequency comparer for (fl3, fl1)
Lfl2	$Lfl2$	Limiter	Frequency comparer for (fl3, fl2)
Lfu1	$Lfu1$	Limiter	Frequency comparer for (fu1, fu3)
Lfu2	$Lfu2$	Limiter	Frequency comparer for (fu2, fu3)
IAWfl1_lim	lim_{IAWfl1}	AntiWindup	Limiter in integrator
IAWfl2_lim	lim_{IAWfl2}	AntiWindup	Limiter in integrator
IAWfu1_lim	lim_{IAWfu1}	AntiWindup	Limiter in integrator
IAWfu2_lim	lim_{IAWfu2}	AntiWindup	Limiter in integrator
LVl1	$LVl1$	Limiter	Voltage comparer for (vl4, vl1)
LVl2	$LVl2$	Limiter	Voltage comparer for (vl4, vl2)
LVl3	$LVl3$	Limiter	Voltage comparer for (vl4, vl3)
LVu1	$LVu1$	Limiter	Voltage comparer for (vu1, vu3)
LVu2	$LVu2$	Limiter	Voltage comparer for (vu2, vu3)
IAWVl1_lim	lim_{IAWVl1}	AntiWindup	Limiter in integrator
IAWVl2_lim	lim_{IAWVl2}	AntiWindup	Limiter in integrator
IAWVl3_lim	lim_{IAWVl3}	AntiWindup	Limiter in integrator
IAWVu1_lim	lim_{IAWVu1}	AntiWindup	Limiter in integrator
IAWVu2_lim	lim_{IAWVu2}	AntiWindup	Limiter in integrator

Blocks

Name	Symbol	Type	Info
IAWfl1	$IAWfl1$	IntegratorAntiWindup	condition check for (fl3, fl1)
IAWfl2	$IAWfl2$	IntegratorAntiWindup	condition check for (fl3, fl2)
IAWfu1	$IAWfu1$	IntegratorAntiWindup	condition check for (fu1, fu3)
IAWfu2	$IAWfu2$	IntegratorAntiWindup	condition check for (fu2, fu3)
IAWVl1	$IAWVl1$	IntegratorAntiWindup	condition check for (Vl3, Vl1)
IAWVl2	$IAWVl2$	IntegratorAntiWindup	condition check for (Vl3, Vl2)
IAWVl3	$IAWVl3$	IntegratorAntiWindup	condition check for (Vl3, Vl2)
IAWVu1	$IAWVu1$	IntegratorAntiWindup	condition check for (Vu1, Vu3)
IAWVu2	$IAWVu2$	IntegratorAntiWindup	condition check for (Vu2, Vu3)

Config Fields in [DGPRCT1]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

DGPRCTExt

DGPRCT External model, follow IEEE-1547-2018. DGPRCT stands for DG protection.

Similar to DGPRCT1, but the measured voltage can be manipulated.

A demo is provided: examples/demonstration/1.2 demo_DGPRCTExt.ipynb

This model can be applied to co-simulation, where you can input the external votage signal into ANDES. If no extertal value is applied, the votalge will remain as the initialized value.

Target device (limited to DG group) `Psum` and `Qsum` will decrease to zero immediately when frequency/voltage protection flag is raised. Once the lock is released, `Psum` and `Qsum` will return to normal immediately.

DG group base model `PVD1` already has a degrading function which is used to degrade output under abnormal condition. it is recommended to turn it off by setting *recflag = 0*.

`fen` and `Ven` are protection enabling parameters. 1/0 is on/off.

`lock` is lock flag signal.

It should be noted that, the lock only lock the `fHz` (frequency read value) of DG model. The source values (which come from `BusFreq` f remain unchanged.)

Protection sensors (e.g., IAWfl1) are instances of `IntergratorAntiWindup`. All the protection sensors will be reset after `lock` returns to 0. Resetting action takes *Tres* to finish.

The model does not check the shedding points sequence. The input parameters are required to satisfy `fl3 < fl2 < fl1 < fu1 < fu2 < fu3`, and `ul4 < ul3 < ul2 < ul1 < uu1 < uu2 < uu3`.

Default settings:

Frequency (Hz):

(fl3, fl2), Tfl2 [(50.0, 57.5), 10s]

(fl2, fl1), Tfl1 [(57.5, 59.2), 300s]

(fu1, fu2), Tfu1 [(60.5, 61.5), 300s]

(fu2, fu3), Tfu2 [(61.5, 70.0), 10s]

Voltage (p.u.):

(vl4, vl3), Tvl3 [(0.10, 0.45), 0.16s]

(vl3, vl2), Tvl2 [(0.45, 0.60), 1s]

(vl2, vl1), Tvl1 [(0.60, 0.88), 2s]

(vu1, vu2), Tvu1 [(1.10, 1.20), 1s]

(vu2, vu3), Tvu2 [(1.20, 2.00), 0.16s]

Reference:

NERC. Bulk Power System Reliability Perspectives on the Adoption of IEEE 1547-2018. March 2020. Available:

https://www.nerc.com/comm/PC_Reliability_Guidelines_DL/Guideline_IEEE_1547-2018_BPS_Perspectives.pdf

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
dev		idx of the target device			mandatory
busfreq		Target device interface bus measurement device idx			
fen	fen	Frequency deviation protection enable. 1 for enable, 0 for disable.	1		
Ven	Ven	Voltage deviation protection enable. 1 for enable, 0 for disable.	0		
fl3	$fl3$	Under frequency shadding point 3	50	*Hz*	
fl2	$fl2$	Over frequency shadding point 2	57.500	*Hz*	
fl1	$fl1$	Under frequency shadding point 1	59.200	*Hz*	
fu1	$fu1$	Over frequency shadding point 1	60.500	*Hz*	
fu2	$fu2$	Over frequency shadding point 2	61.500	*Hz*	
fu3	$fu3$	Over frequency shadding point 3	70	*Hz*	
Tfl1	T_{fl1}	Stand time for (fl2, fl1)	300		non_negative

continues on next page

Table 7 – continued from previous page

Name	Symbol	Description	Default	Unit	Properties
Tfl2	T_{fl2}	Stand time for (fl3, fl2)	10		non_negative
Tfu1	T_{fu1}	Stand time for (fu1, fu2)	300		non_negative
Tfu2	T_{fu2}	Stand time for (fu2, fu3)	10		non_negative
vl4	$vl4$	Under voltage shadding point 4	0.100	*p.u.*	
vl3	$vl3$	Under voltage shadding point 3	0.450	*p.u.*	
vl2	$vl2$	Under voltage shadding point 2	0.600	*p.u.*	
vl1	$vl1$	Under voltage shadding point 1	0.880	*p.u.*	
vu1	$vu1$	Over voltage shadding point 1	1.100	*p.u.*	
vu2	$vu2$	Over voltage shadding point 2	1.200	*p.u.*	
vu3	$vu3$	Over voltage shadding point 3	2	*p.u.*	
Tvl1	T_{vl1}	Stand time for (vl2, vl1)	2		non_negative
Tvl2	T_{vl2}	Stand time for (vl3, vl2)	1		non_negative
Tvl3	T_{vl3}	Stand time for (vl4, vl3)	0.160		non_negative
Tvu1	T_{vu1}	Stand time for (vu1, vu2)	1		non_negative
Tvu2	T_{vu2}	Stand time for (vu2, vu3)	0.160		non_negative
Tres		Integrator reset time	0.050		
ue	u_e	effective online status	0		
bus			0		
fn			0		

Variables

Name	Symbol	Type	Description	Unit	Properties
IAWfl1_y	$IAWfl_{1y}$	State	AW Integrator output		v_str
IAWfl2_y	$IAWfl_{2y}$	State	AW Integrator output		v_str
IAWfu1_y	$IAWfu_{1y}$	State	AW Integrator output		v_str
IAWfu2_y	$IAWfu_{2y}$	State	AW Integrator output		v_str
IAWVl1_y	$IAWVl_{1y}$	State	AW Integrator output		v_str
IAWVl2_y	$IAWVl_{2y}$	State	AW Integrator output		v_str
IAWVl3_y	$IAWVl_{3y}$	State	AW Integrator output		v_str
IAWVu1_y	$IAWVu_{1y}$	State	AW Integrator output		v_str
IAWVu2_y	$IAWVu_{2y}$	State	AW Integrator output		v_str
fHz	fHz	Algeb	frequency in Hz		v_str
dsum	$dsum$	Algeb	lock signal summation		v_str
lock	$lock$	Algeb	lock flag		v_str
f	f	ExtAlgeb	DG frequency read value	*p.u.*	
fin	fin	ExtAlgeb	original f from DG		
fHzl	$fHzl$	ExtAlgeb	Frequency measure lock		
Pext	$Pext$	ExtAlgeb	original Pext from DG		
Pref	$Pref$	ExtAlgeb	original Pref from DG		
Pdrp	$Pdrp$	ExtAlgeb	original Pdrp from DG		
Psum	$Psum$	ExtAlgeb	Active power lock		
Qdrp	$Qdrp$	ExtAlgeb	original Qdrp from DG		
Qref	$Qref$	ExtAlgeb	original Qref from DG		
Qsum	$Qsum$	ExtAlgeb	Reactive power lock		

Initialization Equations

Name	Symbol	Type	Initial Value
IAWfl1_y	$IAWfl_{1y}$	State	0
IAWfl2_y	$IAWfl_{2y}$	State	0
IAWfu1_y	$IAWfu_{1y}$	State	0
IAWfu2_y	$IAWfu_{2y}$	State	0
IAWVl1_y	$IAWVl_{1y}$	State	0
IAWVl2_y	$IAWVl_{2y}$	State	0
IAWVl3_y	$IAWVl_{3y}$	State	0
IAWVu1_y	$IAWVu_{1y}$	State	0
IAWVu2_y	$IAWVu_{2y}$	State	0
fHz	fHz	Algeb	ffn
dsum	$dsum$	Algeb	0
lock	$lock$	Algeb	0
f	f	ExtAlgeb	
fin	fin	ExtAlgeb	
fHzl	$fHzl$	ExtAlgeb	
Pext	$Pext$	ExtAlgeb	
Pref	$Pref$	ExtAlgeb	
Pdrp	$Pdrp$	ExtAlgeb	
Psum	$Psum$	ExtAlgeb	
Qdrp	$Qdrp$	ExtAlgeb	
Qref	$Qref$	ExtAlgeb	
Qsum	$Qsum$	ExtAlgeb	

Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
IAWfl1_y	$IAWfl_{1y}$	State	$Lfl_{1zi}\left(1-res\right)-\frac{Tfl_{1}res}{Tres}$	1
IAWfl2_y	$IAWfl_{2y}$	State	$Lfl_{2zi}\left(1-res\right)-\frac{Tfl_{2}res}{Tres}$	1
IAWfu1_y	$IAWfu_{1y}$	State	$Lfu_{1zi}\left(1-res\right)-\frac{Tfu_{1}res}{Tres}$	1
IAWfu2_y	$IAWfu_{2y}$	State	$Lfl_{2zi}\left(1-res\right)-\frac{Tfu_{2}res}{Tres}$	1
IAWVl1_y	$IAWVl_{1y}$	State	$LVl_{1zi}\left(1-res\right)-\frac{Tvl_{1}res}{Tres}$	1
IAWVl2_y	$IAWVl_{2y}$	State	$LVl_{2zi}\left(1-res\right)-\frac{Tvl_{2}res}{Tres}$	1
IAWVl3_y	$IAWVl_{3y}$	State	$LVl_{2zi}\left(1-res\right)-\frac{Tvl_{3}res}{Tres}$	1
IAWVu1_y	$IAWVu_{1y}$	State	$LVu_{1zi}\left(1-res\right)-\frac{Tvu_{1}res}{Tres}$	1
IAWVu2_y	$IAWVu_{2y}$	State	$LVu_{2zi}\left(1-res\right)-\frac{Tvu_{2}res}{Tres}$	1

Algebraic Equations

Nam	Sym bol	Type	RHS of Equation "0 = g(x, y)"
fHz	fHz	Al-geb	$ffn - fHz$
dsum	$dsun$	Al-geb	$Ven\left(IAWVl_{1limzu}LVl_{1zi} + IAWVl_{2limzu}LVl_{2zi} + IAWVl_{3limzu}LVl_{3zi} + IAWVu_{1limzu}LVu_{1}$ $dsum + fen\left(IAWfl_{1limzu}Lfl_{1zi} + IAWfl_{2limzu}Lfl_{2zi} + IAWfu_{1limzu}Lfu_{1zi} + IAWfu_{2limzu}L$
lock	$lock$	Al-geb	$Ldsum_{zu} - lock$
f	f	Ex-tAl-geb	0
fin	fin	Ex-tAl-geb	0
fHzl	fHz	Ex-tAl-geb	$-ffnlock$
Pext	$Pext$	Ex-tAl-geb	0
Pref	$Prej$	Ex-tAl-geb	0
Pdrp	$Pdrp$	Ex-tAl-geb	0
Psum	$Psun$	Ex-tAl-geb	$-lock\left(Pdrp + Pext + Pref\right)$
Qdrp	$Qdrp$	Ex-tAl-geb	0
Qref	$Qrej$	Ex-tAl-geb	0
Qsun	$Qsun$	Ex-tAl-geb	$-lock\left(Qdrp + Qref\right)$

Services

Name	Symbol	Equation	Type
ltu	ltu	0.8	ConstService
ltl	ltl	0.2	ConstService
zero	$zero$	0	ConstService
res	res	0	ExtendedEvent

Discretes

Name	Symbol	Type	Info
Ldsum	$Ldsum$	Limiter	lock signal comparer, zu is to act
Lfl1	$Lfl1$	Limiter	Frequency comparer for (fl3, fl1)
Lfl2	$Lfl2$	Limiter	Frequency comparer for (fl3, fl2)
Lfu1	$Lfu1$	Limiter	Frequency comparer for (fu1, fu3)
Lfu2	$Lfu2$	Limiter	Frequency comparer for (fu2, fu3)
IAWfl1_lim	lim_{IAWfl1}	AntiWindup	Limiter in integrator
IAWfl2_lim	lim_{IAWfl2}	AntiWindup	Limiter in integrator
IAWfu1_lim	lim_{IAWfu1}	AntiWindup	Limiter in integrator
IAWfu2_lim	lim_{IAWfu2}	AntiWindup	Limiter in integrator
LVl1	$LVl1$	Limiter	Voltage comparer for (vl4, vl1)
LVl2	$LVl2$	Limiter	Voltage comparer for (vl4, vl2)
LVl3	$LVl3$	Limiter	Voltage comparer for (vl4, vl3)
LVu1	$LVu1$	Limiter	Voltage comparer for (vu1, vu3)
LVu2	$LVu2$	Limiter	Voltage comparer for (vu2, vu3)
IAWVl1_lim	lim_{IAWVl1}	AntiWindup	Limiter in integrator
IAWVl2_lim	lim_{IAWVl2}	AntiWindup	Limiter in integrator
IAWVl3_lim	lim_{IAWVl3}	AntiWindup	Limiter in integrator
IAWVu1_lim	lim_{IAWVu1}	AntiWindup	Limiter in integrator
IAWVu2_lim	lim_{IAWVu2}	AntiWindup	Limiter in integrator

Blocks

Name	Symbol	Type	Info
IAWfl1	$IAWfl1$	IntegratorAntiWindup	condition check for (fl3, fl1)
IAWfl2	$IAWfl2$	IntegratorAntiWindup	condition check for (fl3, fl2)
IAWfu1	$IAWfu1$	IntegratorAntiWindup	condition check for (fu1, fu3)
IAWfu2	$IAWfu2$	IntegratorAntiWindup	condition check for (fu2, fu3)
IAWVl1	$IAWVl1$	IntegratorAntiWindup	condition check for (Vl3, Vl1)
IAWVl2	$IAWVl2$	IntegratorAntiWindup	condition check for (Vl3, Vl2)
IAWVl3	$IAWVl3$	IntegratorAntiWindup	condition check for (Vl3, Vl2)
IAWVu1	$IAWVu1$	IntegratorAntiWindup	condition check for (Vu1, Vu3)
IAWVu2	$IAWVu2$	IntegratorAntiWindup	condition check for (Vu2, Vu3)

Config Fields in [DGPRCTExt]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

5.3.10 DataSeries

Group for TimeSeries models.

Available models: *TimeSeries*

TimeSeries

Model for applying time-series data.

A TimeSeries device takes a *xlsx* data spreadsheet and applies the data to the specified device. The spreadsheet can contain multiple sheets, each with a column named `t` and multiple user-defined columns for the data. The values will be applied at the exact time instant.

The `xlsx` data spreadsheet is assumed in the same folder as the case file.

Regarding the parameters for the `TimeSeries` device:

- The column names in the `xlsx` data file need to be specified through the `fields` parameter, separated by commas.
- The parameter/service names of the device which is to be updated need to be specified through the `dests` parameter, separated by commas.

There are a few caveats with the current TimeSeries implementation:

- TimeSeries will not be applied power flow.

- The interpolation mode has yet to be implemented.

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
mode		Mode for applying timeseries. 1: exact time, 2: interpolated	1		
path		Path to timeseries xlsx file.			mandatory
sheet		Sheet name to use			mandatory
fields		comma-separated field names in timeseries data			mandatory
tkey		Key for timestamps	t		
model		Model to link to			mandatory
dev		Idx of device to link to			mandatory
dests		comma-separated device fields as destinations			mandatory
ue	u_e	effective online status	0		

Discretes

Name	Symbol	Type	Info
SW	SW	Switcher	mode switcher

Config Fields in [TimeSeries]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

5.3.11 DynLoad

Dynamic load group.

Common Parameters: u, name

Available models: *ZIP*, *FLoad*

ZIP

ZIP load model (polynomial load). This model is initialized after power flow.

Please check the config of PQ to avoid double counting. If this ZIP model is in use, one should typically set *p2p=1.0* and *q2q=1.0* while leaving the others (*p2i*, *p2z*, *q2i*, *q2z*, and *pq2z*) as zeros. This setting allows one to impose the desired powers by the static PQ and to convert them based on the percentage specified in the ZIP.

The percentages for active power, (*kpp*, *kpi*, and *kpz*) must sum up to 100. Otherwise, initialization will fail. The same applies to the reactive power percentages.

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
pq		idx of the PQ to replace			mandatory
kpp	K_{pp}	Percentage of active power			mandatory
kpi	K_{pi}	Percentage of active current			mandatory
kpz	K_{pz}	Percentage of conductance			mandatory
kqp	K_{qp}	Percentage of reactive power			mandatory
kqi	K_{qi}	Percentage of reactive current			mandatory
kqz	K_{qz}	Percentage of susceptance			mandatory
ue	u_e	effective online status	0		
bus		retrieved bux idx	0		

Variables

Name	Symbol	Type	Description	Unit	Properties
a	a	ExtAlgeb			
v	v	ExtAlgeb			

Initialization Equations

Name	Symbol	Type	Initial Value
a	a	ExtAlgeb	
v	v	ExtAlgeb	

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
a	a	ExtAlgeb	$\pi_0 v + pp_0 + pz_0 v^2$
v	v	ExtAlgeb	$qi_0 v + qp_0 + qz_0 v^2$

Services

Name	Symbol	Equation	Type
kps	K_{psum}	$kpi + kpp + kpz$	ConstService
kqs	K_{qsum}	$kqi + kqp + kqz$	ConstService
rpp	r_{pp}	$\frac{kppu}{100}$	ConstService
rpi	r_{pi}	$\frac{kpiu}{100}$	ConstService
rpz	r_{pz}	$\frac{kpzu}{100}$	ConstService
rqp	r_{qp}	$\frac{kqpu}{100}$	ConstService
rqi	r_{qi}	$\frac{kqiu}{100}$	ConstService
rqz	r_{qz}	$\frac{kqzu}{100}$	ConstService
pp0	P_{p0}	$p_0 rpp$	ConstService
pi0	P_{i0}	$\frac{p_0 rpi}{v_0}$	ConstService
pz0	P_{z0}	$\frac{p_0 rpz}{v_0^2}$	ConstService
qp0	Q_{p0}	$q_0 rqp$	ConstService
qi0	Q_{i0}	$\frac{q_0 rqi}{v_0}$	ConstService
qz0	Q_{z0}	$\frac{q_0 rqz}{v_0^2}$	ConstService

Config Fields in [ZIP]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

FLoad

Voltage and frequency dependent load.

FLoad replaces the linked PQ load (set to u=0) and computes power as `P = pv0 * v^ap * f^bp`, `Q = qv0 * v^aq * f^bq`.

`pv0` and `qv0` are ConstService values computed once at TDS init from the PQ's solved power flow values (`Ppf`, `Qpf`). To adjust power injection during TDS, alter `pv0`/`qv0` on FLoad directly:

```
Alter(model='FLoad', src='pv0', dev=fl_idx, t=5, method='+', amount=0.1)
```

Altering the linked PQ's `Ppf` has no effect since PQ is disabled.

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
pq		idx of the PQ to replace			mandatory
busf		optional idx of the BusFreq device to use			
kp		active power percentage	100	%	
kq		active power percentage	100	%	
Tf		filter time constant	0.020	*s*	non_negative
ap		active power voltage exponent	1		
aq		reactive power voltage exponent	0		
bp		active power frequency exponent	0		
bq		reactive power frequency exponent	0		
ue	u_e	effective online status	0		
bus			0		

Variables

Name	Symbol	Type	Description	Unit	Properties
f	f	ExtAlgeb			
a	a	ExtAlgeb			
v	v	ExtAlgeb			

Initialization Equations

Name	Symbol	Type	Initial Value
f	f	ExtAlgeb	
a	a	ExtAlgeb	
v	v	ExtAlgeb	

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
f	f	ExtAlgeb	0
a	a	ExtAlgeb	$f^{bp} pv_0 v^{ap}$
v	v	ExtAlgeb	$f^{bq} qv_0 v^{aq}$

Services

Name	Symbol	Equation	Type
pv0	$pv0$	$\frac{kpp_0 uv_0^{-ap}}{100}$	ConstService
qv0	$qv0$	$\frac{kqq_0 uv_0^{-aq}}{100}$	ConstService

Config Fields in [FLoad]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

5.3.12 DynShaft

Dynamic shaft model group for multi-mass torsional models.

Common Parameters: u, name, syn

Available models: *SHAFT5*

SHAFT5

5-mass torsional shaft model (HP-IP-LP-Rotor-EX) for synchronous generators.

The model adds torsional dynamics to a synchronous generator by introducing 4 additional mass-spring-damper sections connected in series with the generator rotor.

The 5 masses represent:

- HP: High-pressure turbine section
- IP: Intermediate-pressure turbine section
- LP: Low-pressure turbine section
- Rotor: Generator rotor (existing SynGen omega/delta)
- EX: Exciter mass

The mechanical torque (tm) from the governor drives the HP mass. The generator's omega equation is modified to remove tm and add shaft coupling torques from LP and EX masses.

At steady state, all mass speeds equal 1.0 p.u. and the spring torques balance the mechanical torque through the chain.

References

Milano, F. (2010). Power System Modelling and Scripting, Section 15.1.10.

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
syn		Synchronous generator idx			mandatory,unique
MHP	M_{HP}	HP mass start-up time (2*H_HP)	0.092	*MWs/MVA*	non_zero,non_negative,power
MIP	M_{IP}	IP mass start-up time (2*H_IP)	0.156	*MWs/MVA*	non_zero,non_negative,power
MLP	M_{LP}	LP mass start-up time (2*H_LP)	0.858	*MWs/MVA*	non_zero,non_negative,power
MEX	M_{EX}	EX mass start-up time (2*H_EX)	0.068	*MWs/MVA*	non_zero,non_negative,power
KHP	K_{HP}	HP-IP spring constant	19.303	*p.u.*	non_zero,non_negative,power
KIP	K_{IP}	IP-LP spring constant	34.929	*p.u.*	non_zero,non_negative,power
KLP	K_{LP}	LP-Rotor spring constant	52.038	*p.u.*	non_zero,non_negative,power
KEX	K_{EX}	Rotor-EX spring constant	1.550	*p.u.*	non_zero,non_negative,power
DHP	D_{HP}	HP mass self-damping	0	*p.u.*	power
DIP	D_{IP}	IP mass self-damping	0	*p.u.*	power
DLP	D_{LP}	LP mass self-damping	0	*p.u.*	power
DEX	D_{EX}	EX mass self-damping	0	*p.u.*	power
D12	D_{12}	HP-IP mutual damping	0	*p.u.*	power
D23	D_{23}	IP-LP mutual damping	0	*p.u.*	power
D34	D_{34}	LP-Rotor mutual damping	0	*p.u.*	power
D45	D_{45}	Rotor-EX mutual damping	0	*p.u.*	power
ue	u_e	effective online status	0		
Sn	S_n	Generator power rating	0		
fn	f_n	Rated frequency	0		

Variables

Name	Symbol	Type	Description	Unit	Properties
dHP	dHP	State	HP turbine angle	*rad*	v_str
wHP	wHP	State	HP turbine speed	*p.u.*	v_str
dIP	dIP	State	IP turbine angle	*rad*	v_str
wIP	wIP	State	IP turbine speed	*p.u.*	v_str
dLP	dLP	State	LP turbine angle	*rad*	v_str
wLP	wLP	State	LP turbine speed	*p.u.*	v_str
dEX	dEX	State	Exciter mass angle	*rad*	v_str
wEX	wEX	State	Exciter mass speed	*p.u.*	v_str
omega	ω	ExtState	Generator rotor speed		
delta	δ	ExtState	Generator rotor angle		
tm	tm	ExtAlgeb	Mechanical torque from generator		

Initialization Equations

Name	Symbol	Type	Initial Value
dHP	dHP	State	$\delta_{0gen} + \frac{tm_0}{KLP} + \frac{tm_0}{KIP} + \frac{tm_0}{KHP}$
wHP	wHP	State	1
dIP	dIP	State	$\delta_{0gen} + \frac{tm_0}{KLP} + \frac{tm_0}{KIP}$
wIP	wIP	State	1
dLP	dLP	State	$\delta_{0gen} + \frac{tm_0}{KLP}$
wLP	wLP	State	1
dEX	dEX	State	δ_{0gen}
wEX	wEX	State	1
omega	ω	ExtState	
delta	δ	ExtState	
tm	tm	ExtAlgeb	

Differential Equations

Nam	Sym-bol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
dHP	dHP	State	$Obue\left(wHP - 1\right)$	
wHP	wHP	State	$ue\left(-D_{12}\left(wHP - wIP\right) - DHP\left(wHP - 1\right) + KHP\left(-dHP + dIP\right)\right.$	M_{HP}
dIP	dIP	State	$Obue\left(wIP - 1\right)$	
wIP	wIP	State	$ue\left(-D_{12}\left(-wHP + wIP\right) - D_{23}\left(wIP - wLP\right) - DIP\left(wIP - 1\right) + \right.$	M_{IP} $lHP - dIP) +$
dLP	dLP	State	$Obue\left(wLP - 1\right)$	
wLP	wLP	State	$ue\left(-D_{23}\left(-wIP + wLP\right) - D_{34}\left(-\omega + wLP\right) - DLP\left(wLP - 1\right) + K\right.$	M_{LP} $P - dLP) + KI$
dEX	dEX	State	$Obue\left(wEX - 1\right)$	
wEX	wEX	State	$ue\left(-D_{45}\left(-\omega + wEX\right) - DEX\left(wEX - 1\right) + KEX\left(-dEX + \delta\right)\right)$	M_{EX}
omeg	ω	ExtSta	$ue\left(-D_{34}\left(\omega - wLP\right) - D_{45}\left(\omega - wEX\right) + KEX\left(dEX - \delta\right) + KLP\left(\right.\right.$	$\delta) - tm)$
delta	δ	ExtSta	0	

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
tm	tm	ExtAlgeb	0

Services

Name	Symbol	Equation	Type
Ob	Ω_b	$2\pi fn$	ConstService

Config Fields in [SHAFT5]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

5.3.13 Exciter

Exciter group for synchronous generators.

Common Parameters: u, name, syn

Common Variables: vout, vi

Available models: *EXDC2*, *IEEEX1*, *ESDC1A*, *ESDC2A*, *EXST1*, *ESST3A*, *SEXS*, *IEEET1*, *EXAC1*, *EXAC2*, *EXAC4*, *ESST4B*, *AC8B*, *IEEET3*, *ESAC1A*, *ESST1A*, *ESAC5A*

EXDC2

EXDC2 model.

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
syn		Synchronous generator idx			mandatory
TR	T_R	Sensing time constant	0.010	*p.u.*	
TA	T_A	Lag time constant in anti-windup lag	0.040	*p.u.*	
TC	T_C	Lead time constant in lead-lag	1	*p.u.*	
TB	T_B	Lag time constant in lead-lag	1	*p.u.*	
TE	T_E	Exciter integrator time constant	0.800	*p.u.*	
TF1	T_{F1}	Feedback washout time constant	1	*p.u.*	non_zero
KF1	K_{F1}	Feedback washout gain	0.030	*p.u.*	
KA	K_A	Gain in anti-windup lag TF	40	*p.u.*	
KE	K_E	Gain added to saturation	1	*p.u.*	
VRMAX	V_{RMAX}	Maximum excitation limit	7.300	*p.u.*	
VRMIN	V_{RMIN}	Minimum excitation limit	-7.300	*p.u.*	
E1	E_1	First saturation point	0	*p.u.*	
SE1	S_{E1}	Value at first saturation point	0	*p.u.*	
E2	E_2	Second saturation point	1	*p.u.*	
SE2	S_{E2}	Value at second saturation point	1	*p.u.*	
ue	u_e	effective online status	0		
Sn	S_m	Rated power from generator	0	*MVA*	
Vn	V_m	Rated voltage from generator	0	*kV*	
bus	bus	Bus idx of the generators	0		

Variables

Name	Symbol	Type	Description	Unit	Properties
vp	vp	State	Voltage after saturation feedback, before speed term	*p.u.*	v_str
LS_y	LS_y	State	State in lag transfer function		v_str
LL_x	LL_x	State	State in lead-lag		v_str
LA_y	LA_y	State	State in lag TF		v_str
W_x	W_x	State	State in washout filter		v_str
omega	ω	ExtState	Generator speed		
v	v	Algeb	Input to exciter (bus v or Eterm)		v_str
vout	$vout$	Algeb	Exciter final output voltage		v_str
vref	$vref$	Algeb	Reference voltage input	*p.u.*	v_str
Se	Se	Algeb	saturation output		v_str
vi	vi	Algeb	Total input voltages	*p.u.*	v_str
LL_y	LL_y	Algeb	Output of lead-lag		v_str
W_y	W_y	Algeb	Output of washout filter		v_str
vf	vf	ExtAlgeb	Excitation field voltage to generator		
XadIfd	$XadIfd$	ExtAlgeb	Armature excitation current		
a	a	ExtAlgeb	Bus voltage phase angle		
vbus	$vbus$	ExtAlgeb	Bus voltage magnitude		

Initialization Equations

Name	Symbol	Type	Initial Value
vp	vp	State	vf_0
LS_y	LS_y	State	$1.0v$
LL_x	LL_x	State	vi
LA_y	LA_y	State	$KALL_y$
W_x	W_x	State	vp
omega	ω	ExtState	
v	v	Algeb	$vbus$
vout	$vout$	Algeb	$uevf_0$
vref	$vref$	Algeb	$v + vb_0$
Se	Se	Algeb	Se_0
vi	vi	Algeb	vb_0
LL_y	LL_y	Algeb	vi
W_y	W_y	Algeb	0
vf	vf	ExtAlgeb	
XadIfd	$XadIfd$	ExtAlgeb	
a	a	ExtAlgeb	
vbus	$vbus$	ExtAlgeb	

Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
vp	vp	State	$ue\left(-KEvp + LA_y - Sevp\right)$	T_E
LS_y	LS_y	State	$-LS_y + 1.0v$	T_R
LL_x	LL_x	State	$-LL_x + vi$	T_B
LA_y	LA_y	State	$KALL_y - LA_y$	T_A
W_x	W_x	State	$-W_x + vp$	T_{F1}
omega	ω	ExtState	0	

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
v	v	Algeb	$-v + vbus$
vout	$vout$	Algeb	$\omega uevp - vout$
vref	$vref$	Algeb	$-vref + vref_0$
Se	Se	Algeb	$SAT_B SL_{z0} \left(-SAT_A + vp\right)^2 - Sevp$
vi	vi	Algeb	$-LS_y - W_y - vi + vref$
LL_y	LL_y	Algeb	$LL_{LTz1} \left(LL_y - vi\right) + LL_x TB - LL_y TB + TC \left(-LL_x + vi\right)$
W_y	W_y	Algeb	$KF_1 \left(-W_x + vp\right) - TF_1 W_y$
vf	vf	ExtAlgeb	$ue \left(-vf_0 + vout\right)$
XadIfd	$XadIfd$	ExtAlgeb	0
a	a	ExtAlgeb	0
vbus	$vbus$	ExtAlgeb	0

Services

Name	Symbol	Equation	Type
SAT_E1	E_{SAT}^{1c}	E_1	ConstService
SAT_E2	E_{SAT}^{2c}	$E_2 - 2SAT_{zSE2} + 2$	ConstService
SAT_SE1	SE_{SAT}^{1c}	SE_1	ConstService
SAT_SE2	SE_{SAT}^{2c}	$-2SAT_{zSE2} + SE_2 + 2$	ConstService
SAT_a	a_{SAT}	$\sqrt{\frac{SAT_{E1} SAT_{SE1}}{SAT_{E2} SAT_{SE2}}} \left(\text{Indicator}\left(SAT_{SE2} > 0\right) + \text{Indicator}\left(SAT_{SE2} <\right.\right.$	ConstService
SAT_A	A_{SAT}^{q}	$SAT_{E2} - \frac{SAT_{E1} - SAT_{E2}}{SAT_a - 1}$	ConstService
SAT_B	B_{SAT}^{q}	$\frac{SAT_{E2} SAT_{SE2} \left(SAT_a - 1\right)^2 \left(\text{Indicator}\left(SAT_a > 0\right) + \text{Indicator}\left(SAT_a < 0\right)\right)}{\left(SAT_{E1} - SAT_{E2}\right)^2}$	ConstService
Se0	S_{e0}	$\frac{SAT_B \left(SAT_A - vf_0\right)^2 \text{Indicator}\left(vf_0 > SAT_A\right)}{-ue + vf_0 + 1}$	ConstService
vr0	V_{r0}	$vf_0 \left(KE + Se_0\right)$	ConstService
vb0	V_{b0}	$\frac{vr_0}{KA}$	ConstService
vref0	V_{ref0}	$vref$	PostInitService

Discretes

Name	Symbol	Type	Info
SL	SL	LessThan	
LL_LT	LT_{LL}	LessThan	
LA_lim	lim_{LA}	AntiWindup	Limiter in Lag

Blocks

Name	Symbol	Type	Info
SAT	SAT	ExcQuadSat	Field voltage saturation
LS	LS	Lag	Sensing lag TF
LL	LL	LeadLag	Lead-lag for internal delays
LA	LA	LagAntiWindup	Anti-windup lag
W	W	Washout	Signal conditioner

Setpoints

Name	Target	Info
vref	vref0	Constant v ref

Config Fields in [EXDC2]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

IEEEX1

IEEEX1 Type 1 exciter (DC)

Derived from EXDC2 by varying the limiter bounds.

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
syn		Synchronous generator idx			mandatory
TR	T_R	Sensing time constant	0.010	*p.u.*	
TA	T_A	Lag time constant in anti-windup lag	0.040	*p.u.*	
TC	T_C	Lead time constant in lead-lag	1	*p.u.*	
TB	T_B	Lag time constant in lead-lag	1	*p.u.*	
TE	T_E	Exciter integrator time constant	0.800	*p.u.*	
TF1	T_{F1}	Feedback washout time constant	1	*p.u.*	non_zero
KF1	K_{F1}	Feedback washout gain	0.030	*p.u.*	
KA	K_A	Gain in anti-windup lag TF	40	*p.u.*	
KE	K_E	Gain added to saturation	1	*p.u.*	
VRMAX	V_{RMAX}	Maximum excitation limit	7.300	*p.u.*	
VRMIN	V_{RMIN}	Minimum excitation limit	-7.300	*p.u.*	
E1	E_1	First saturation point	0	*p.u.*	
SE1	S_{E1}	Value at first saturation point	0	*p.u.*	
E2	E_2	Second saturation point	1	*p.u.*	
SE2	S_{E2}	Value at second saturation point	1	*p.u.*	
ue	u_e	effective online status	0		
Sn	S_m	Rated power from generator	0	*MVA*	
Vn	V_m	Rated voltage from generator	0	*kV*	
bus	bus	Bus idx of the generators	0		

Variables

Name	Symbol	Type	Description	Unit	Properties
vp	vp	State	Voltage after saturation feedback, before speed term	*p.u.*	v_str
LS_y	LS_y	State	State in lag transfer function		v_str
LL_x	LL_x	State	State in lead-lag		v_str
LA_y	LA_y	State	State in lag TF		v_str
W_x	W_x	State	State in washout filter		v_str
omega	ω	ExtState	Generator speed		
v	v	Algeb	Input to exciter (bus v or Eterm)		v_str
vout	$vout$	Algeb	Exciter final output voltage		v_str
vref	$vref$	Algeb	Reference voltage input	*p.u.*	v_str
Se	Se	Algeb	saturation output		v_str
vi	vi	Algeb	Total input voltages	*p.u.*	v_str
LL_y	LL_y	Algeb	Output of lead-lag		v_str
W_y	W_y	Algeb	Output of washout filter		v_str
vf	vf	ExtAlgeb	Excitation field voltage to generator		
XadIfd	$XadIfd$	ExtAlgeb	Armature excitation current		
a	a	ExtAlgeb	Bus voltage phase angle		
vbus	$vbus$	ExtAlgeb	Bus voltage magnitude		

Initialization Equations

Name	Symbol	Type	Initial Value
vp	vp	State	vf_0
LS_y	LS_y	State	$1.0v$
LL_x	LL_x	State	vi
LA_y	LA_y	State	$KALL_y$
W_x	W_x	State	vp
omega	ω	ExtState	
v	v	Algeb	$vbus$
vout	$vout$	Algeb	$uevf_0$
vref	$vref$	Algeb	$v + vb_0$
Se	Se	Algeb	Se_0
vi	vi	Algeb	vb_0
LL_y	LL_y	Algeb	vi
W_y	W_y	Algeb	0
vf	vf	ExtAlgeb	
XadIfd	$XadIfd$	ExtAlgeb	
a	a	ExtAlgeb	
vbus	$vbus$	ExtAlgeb	

Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
vp	vp	State	$ue\left(-KEvp + LA_y - Sevp\right)$	T_E
LS_y	LS_y	State	$-LS_y + 1.0v$	T_R
LL_x	LL_x	State	$-LL_x + vi$	T_B
LA_y	LA_y	State	$KALL_y - LA_y$	T_A
W_x	W_x	State	$-W_x + vp$	T_{F1}
omega	ω	ExtState	0	

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
v	v	Algeb	$-v + vbus$
vout	$vout$	Algeb	$uevp - vout$
vref	$vref$	Algeb	$-vref + vref_0$
Se	Se	Algeb	$SAT_B SL_{z0} \left(-SAT_A + vp\right)^2 - Sevp$
vi	vi	Algeb	$-LS_y - W_y - vi + vref$
LL_y	LL_y	Algeb	$LL_{LTz1} \left(LL_y - vi\right) + LL_x TB - LL_y TB + TC \left(-LL_x + vi\right)$
W_y	W_y	Algeb	$KF_1 \left(-W_x + vp\right) - TF_1 W_y$
vf	vf	ExtAlgeb	$ue \left(-vf_0 + vout\right)$
XadIfd	$XadIfd$	ExtAlgeb	0
a	a	ExtAlgeb	0
vbus	$vbus$	ExtAlgeb	0

Services

Name	Symbol	Equation	Type
SAT_E1	E_{SAT}^{1c}	E_1	ConstService
SAT_E2	E_{SAT}^{2c}	$E_2 - 2SAT_{zSE2} + 2$	ConstService
SAT_SE1	SE_{SAT}^{1c}	SE_1	ConstService
SAT_SE2	SE_{SAT}^{2c}	$-2SAT_{zSE2} + SE_2 + 2$	ConstService
SAT_a	a_{SAT}	$\sqrt{\frac{SAT_{E1} SAT_{SE1}}{SAT_{E2} SAT_{SE2}}} \left(\text{Indicator}\left(SAT_{SE2} > 0\right) + \text{Indicator}\left(SAT_{SE2}\right.\right.$	ConstService
SAT_A	A_{SAT}^{q}	$SAT_{E2} - \frac{SAT_{E1} - SAT_{E2}}{SAT_a - 1}$	ConstService
SAT_B	B_{SAT}^{q}	$\frac{SAT_{E2} SAT_{SE2} \left(SAT_a - 1\right)^2 \left(\text{Indicator}\left(SAT_a > 0\right) + \text{Indicator}\left(SAT_a < 0\right)\right)}{\left(SAT_{E1} - SAT_{E2}\right)^2}$	ConstService
Se0	S_{e0}	$\frac{SAT_B \left(SAT_A - vf_0\right)^2 \text{Indicator}\left(vf_0 > SAT_A\right)}{-ue + vf_0 + 1}$	ConstService
vr0	V_{r0}	$vf_0 \left(KE + Se_0\right)$	ConstService
vb0	V_{b0}	$\frac{vr_0}{KA}$	ConstService
vref0	V_{ref0}	$vref$	PostInitService
VRTMAX	$V_{RMAX} V_T$	$VRMAXv$	VarService
VRTMIN	$V_{RMIN} V_T$	$VRMINv$	VarService

Discretes

Name	Symbol	Type	Info
SL	SL	LessThan	
LL_LT	LT_{LL}	LessThan	
LA_lim	lim_{LA}	AntiWindup	Limiter in Lag

Blocks

Name	Symbol	Type	Info
SAT	SAT	ExcQuadSat	Field voltage saturation
LS	LS	Lag	Sensing lag TF
LL	LL	LeadLag	Lead-lag for internal delays
LA	LA	LagAntiWindup	Anti-windup lag
W	W	Washout	Signal conditioner

Setpoints

Name	Target	Info
vref	vref0	Constant v ref

Config Fields in [IEEEX1]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

ESDC1A

ESDC1A model.

This model derives from ESDC2A and changes the regular limits to "VRMAX" and "VRMIN".

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
syn		Synchronous generator idx			mandatory
TR	T_R	Sensing time constant	0.010	*p.u.*	
KA	K_A	Regulator gain	80		
TA	T_A	Lag time constant in regulator	0.040	*p.u.*	
TB	T_B	Lag time constant in lead-lag	1	*p.u.*	
TC	T_C	Lead time constant in lead-lag	1	*p.u.*	
VRMAX	V_{RMAX}	Max. exc. limit (0-unlimited)	7.300	*p.u.*	
VRMIN	V_{RMIN}	Min. excitation limit	-7.300	*p.u.*	
KE	K_E	Saturation feedback gain	1	*p.u.*	
TE	T_E	Integrator time constant	0.800	*p.u.*	
KF	K_F	Feedback gain	0.100		
TF1	T_{F1}	Feedback washout time constant	1	*p.u.*	non_zero,non_negative
Switch	S_w	Switch that PSS/E did not implement	0	*bool*	
E1	E_1	First saturation point	0	*p.u.*	
SE1	S_{E1}	Value at first saturation point	0	*p.u.*	
E2	E_2	Second saturation point	0	*p.u.*	
SE2	S_{E2}	Value at second saturation point	0	*p.u.*	
ue	u_e	effective online status	0		
Sn	S_m	Rated power from generator	0	*MVA*	
Vn	V_m	Rated voltage from generator	0	*kV*	
bus	bus	Bus idx of the generators	0		

Variables

Name	Symbol	Type	Description	Unit	Properties
LG_y	LG_y	State	State in lag transfer function		v_str
LL_x	LL_x	State	State in lead-lag		v_str
LA_y	LA_y	State	State in lag TF		v_str
INT_y	INT_y	State	Integrator output		v_str
WF_x	WF_x	State	State in washout filter		v_str
omega	ω	ExtState	Generator speed		
v	v	Algeb	Input to exciter (bus v or Eterm)		v_str
vout	$vout$	Algeb	Exciter final output voltage		v_str
vref	$vref$	Algeb	Reference voltage input	*p.u.*	v_str
vi	vi	Algeb	Total input voltages	*p.u.*	v_str
LL_y	LL_y	Algeb	Output of lead-lag		v_str
UEL	UEL	Algeb	Interface var for under exc. limiter		v_str
Se	Se	Algeb	saturation output		v_str
VFE	VFE	Algeb	Combined saturation feedback	*p.u.*	v_str
WF_y	WF_y	Algeb	Output of washout filter		v_str
vf	vf	ExtAlgeb	Excitation field voltage to generator		
XadIfd	$XadIfd$	ExtAlgeb	Armature excitation current		
a	a	ExtAlgeb	Bus voltage phase angle		
vbus	$vbus$	ExtAlgeb	Bus voltage magnitude		

Initialization Equations

Name	Symbol	Type	Initial Value
LG_y	LG_y	State	v
LL_x	LL_x	State	vi
LA_y	LA_y	State	$KALL_y$
INT_y	INT_y	State	vf_0
WF_x	WF_x	State	INT_y
omega	ω	ExtState	
v	v	Algeb	$vbus$
vout	$vout$	Algeb	$uevf_0$
vref	$vref$	Algeb	$v + \frac{vfe_0}{KA}$
vi	vi	Algeb	$\frac{vfe_0}{KA}$
LL_y	LL_y	Algeb	vi
UEL	UEL	Algeb	0
Se	Se	Algeb	Se_0
VFE	VFE	Algeb	vfe_0
WF_y	WF_y	Algeb	0
vf	vf	ExtAlgeb	
XadIfd	$XadIfd$	ExtAlgeb	
a	a	ExtAlgeb	
vbus	$vbus$	ExtAlgeb	

Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
LG_y	LG_y	State	$-LG_y + v$	T_R
LL_x	LL_x	State	$-LL_x + vi$	T_B
LA_y	LA_y	State	$KALL_y - LA_y$	T_A
INT_y	INT_y	State	$ue\,(LA_y - VFE)$	T_E
WF_x	WF_x	State	$INT_y - WF_x$	T_{F1}
omega	ω	ExtState	0	

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
v	v	Algeb	$-v + vbus$
vout	$vout$	Algeb	$INT_y - vout$
vref	$vref$	Algeb	$-vref + vref_0$
vi	vi	Algeb	$-WF_y - v - vi + vref$
LL_y	LL_y	Algeb	$LL_{LTz1}\left(LL_y - vi\right) + LL_x TB - LL_y TB + TC\left(-LL_x + vi\right)$
UEL	UEL	Algeb	$-UEL$
Se	Se	Algeb	$SAT_B SL_{z0}\left(INT_y - SAT_A\right)^2 - Se$
VFE	VFE	Algeb	$INT_y KE + Se - VFE$
WF_y	WF_y	Algeb	$KF\left(INT_y - WF_x\right) - TF_1 WF_y$
vf	vf	ExtAlgeb	$ue\left(-vf_0 + vout\right)$
XadIfd	$XadIfd$	ExtAlgeb	0
a	a	ExtAlgeb	0
vbus	$vbus$	ExtAlgeb	0

Services

Name	Symbol	Equation	Type
VRMAXc	$VRMAXc$	$VRMAX - 999_z VRM + 999$	ConstService
SAT_E1	E_{SAT}^{1c}	E_1	ConstService
SAT_E2	E_{SAT}^{2c}	$E_2 - 2SAT_{zSE2} + 2$	ConstService
SAT_SE1	SE_{SAT}^{1c}	SE_1	ConstService
SAT_SE2	SE_{SAT}^{2c}	$-2SAT_{zSE2} + SE_2 + 2$	ConstService
SAT_a	a_{SAT}	$\sqrt{\frac{SAT_{E1}SAT_{SE1}}{SAT_{E2}SAT_{SE2}}}\left(\text{Indicator}\left(SAT_{SE2} > 0\right) + \text{Indicator}\left(SAT_{SE2}\right.\right.$	ConstService
SAT_A	A_{SAT}^q	$SAT_{E2} - \frac{SAT_{E1} - SAT_{E2}}{SAT_a - 1}$	ConstService
SAT_B	B_{SAT}^q	$\frac{SAT_{E2}SAT_{SE2}\left(SAT_a - 1\right)^2\left(\text{Indicator}\left(SAT_a > 0\right) + \text{Indicator}\left(SAT_a < 0\right)\right)}{\left(SAT_{E1} - SAT_{E2}\right)^2}$	ConstService
Se0	S_{e0}	$SAT_B\left(SAT_A - vf_0\right)^2 \text{Indicator}\left(vf_0 > SAT_A\right)$	ConstService
vfe0	V_{FE0}	$KEvf_0 + Se_0$	ConstService
vref0	V_{ref0}	$vref$	PostInitService
VRU	V_{RMAX}	$VRMAXc$	ConstService
VRL	V_{RMIN}	$VRMIN$	ConstService

Discretes

Name	Symbol	Type	Info
LL_LT	LT_{LL}	LessThan	
HG_lt	$None_{HG}$	LessThan	
LA_lim	lim_{LA}	AntiWindup	Limiter in Lag
SL	SL	LessThan	

Blocks

Name	Symbol	Type	Info
LG	LG	Lag	Transducer delay
SAT	SAT	ExcQuadSat	Field voltage saturation
LL	LL	LeadLag	Lead-lag compensator
HG	HG	HVGate	HVGate for under excitation
LA	LA	LagAntiWindup	Anti-windup lag
INT	INT	Integrator	Integrator
WF	WF	Washout	Feedback to input

Setpoints

Name	Target	Info
vref	vref0	Const reference voltage

Config Fields in [ESDC1A]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

ESDC2A

ESDC2A model.

This model is implemented as described in the PSS/E manual, except that the HVGate is not in use. Due to the HVGate and saturation function, the results are close to but different from TSAT.

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
syn		Synchronous generator idx			mandatory
TR	T_R	Sensing time constant	0.010	*p.u.*	
KA	K_A	Regulator gain	80		
TA	T_A	Lag time constant in regulator	0.040	*p.u.*	
TB	T_B	Lag time constant in lead-lag	1	*p.u.*	
TC	T_C	Lead time constant in lead-lag	1	*p.u.*	
VRMAX	V_{RMAX}	Max. exc. limit (0-unlimited)	7.300	*p.u.*	
VRMIN	V_{RMIN}	Min. excitation limit	-7.300	*p.u.*	
KE	K_E	Saturation feedback gain	1	*p.u.*	
TE	T_E	Integrator time constant	0.800	*p.u.*	
KF	K_F	Feedback gain	0.100		
TF1	T_{F1}	Feedback washout time constant	1	*p.u.*	non_zero,non_negative
Switch	S_w	Switch that PSS/E did not implement	0	*bool*	
E1	E_1	First saturation point	0	*p.u.*	
SE1	S_{E1}	Value at first saturation point	0	*p.u.*	
E2	E_2	Second saturation point	0	*p.u.*	
SE2	S_{E2}	Value at second saturation point	0	*p.u.*	
ue	u_e	effective online status	0		
Sn	S_m	Rated power from generator	0	*MVA*	
Vn	V_m	Rated voltage from generator	0	*kV*	
bus	bus	Bus idx of the generators	0		

Variables

Name	Symbol	Type	Description	Unit	Properties
LG_y	LG_y	State	State in lag transfer function		v_str
LL_x	LL_x	State	State in lead-lag		v_str
LA_y	LA_y	State	State in lag TF		v_str
INT_y	INT_y	State	Integrator output		v_str
WF_x	WF_x	State	State in washout filter		v_str
omega	ω	ExtState	Generator speed		
v	v	Algeb	Input to exciter (bus v or Eterm)		v_str
vout	$vout$	Algeb	Exciter final output voltage		v_str
vref	$vref$	Algeb	Reference voltage input	*p.u.*	v_str
vi	vi	Algeb	Total input voltages	*p.u.*	v_str
LL_y	LL_y	Algeb	Output of lead-lag		v_str
UEL	UEL	Algeb	Interface var for under exc. limiter		v_str
Se	Se	Algeb	saturation output		v_str
VFE	VFE	Algeb	Combined saturation feedback	*p.u.*	v_str
WF_y	WF_y	Algeb	Output of washout filter		v_str
vf	vf	ExtAlgeb	Excitation field voltage to generator		
XadIfd	$XadIfd$	ExtAlgeb	Armature excitation current		
a	a	ExtAlgeb	Bus voltage phase angle		
vbus	$vbus$	ExtAlgeb	Bus voltage magnitude		

Initialization Equations

Name	Symbol	Type	Initial Value
LG_y	LG_y	State	v
LL_x	LL_x	State	vi
LA_y	LA_y	State	$KALL_y$
INT_y	INT_y	State	vf_0
WF_x	WF_x	State	INT_y
omega	ω	ExtState	
v	v	Algeb	$vbus$
vout	$vout$	Algeb	$uevf_0$
vref	$vref$	Algeb	$v + \frac{vfe_0}{KA}$
vi	vi	Algeb	$\frac{vfe_0}{KA}$
LL_y	LL_y	Algeb	vi
UEL	UEL	Algeb	0
Se	Se	Algeb	Se_0
VFE	VFE	Algeb	vfe_0
WF_y	WF_y	Algeb	0
vf	vf	ExtAlgeb	
XadIfd	$XadIfd$	ExtAlgeb	
a	a	ExtAlgeb	
vbus	$vbus$	ExtAlgeb	

Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
LG_y	LG_y	State	$-LG_y + v$	T_R
LL_x	LL_x	State	$-LL_x + vi$	T_B
LA_y	LA_y	State	$KALL_y - LA_y$	T_A
INT_y	INT_y	State	$ue\left(LA_y - VFE\right)$	T_E
WF_x	WF_x	State	$INT_y - WF_x$	T_{F1}
omega	ω	ExtState	0	

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
v	v	Algeb	$-v + vbus$
vout	$vout$	Algeb	$INT_y - vout$
vref	$vref$	Algeb	$-vref + vref_0$
vi	vi	Algeb	$-WF_y - v - vi + vref$
LL_y	LL_y	Algeb	$LL_{LTz1}\left(LL_y - vi\right) + LL_x TB - LL_y TB + TC\left(-LL_x + vi\right)$
UEL	UEL	Algeb	$-UEL$
Se	Se	Algeb	$SAT_B SL_{z0}\left(INT_y - SAT_A\right)^2 - Se$
VFE	VFE	Algeb	$INT_y KE + Se - VFE$
WF_y	WF_y	Algeb	$KF\left(INT_y - WF_x\right) - TF_1 WF_y$
vf	vf	ExtAlgeb	$ue\left(-vf_0 + vout\right)$
XadIfd	$XadIfd$	ExtAlgeb	0
a	a	ExtAlgeb	0
vbus	$vbus$	ExtAlgeb	0

Services

Name	Symbol	Equation	Type
VRMAXc	$VRMAXc$	$VRMAX - 999_z VRM + 999$	ConstService
SAT_E1	E_{SAT}^{1c}	E_1	ConstService
SAT_E2	E_{SAT}^{2c}	$E_2 - 2SAT_{zSE2} + 2$	ConstService
SAT_SE1	SE_{SAT}^{1c}	SE_1	ConstService
SAT_SE2	SE_{SAT}^{2c}	$-2SAT_{zSE2} + SE_2 + 2$	ConstService
SAT_a	a_{SAT}	$\sqrt{\frac{SAT_{E1}SAT_{SE1}}{SAT_{E2}SAT_{SE2}}}\left(\text{Indicator}\left(SAT_{SE2} > 0\right) + \text{Indicator}\left(SAT_{SE2}\right.\right.$	ConstService
SAT_A	A_{SAT}^q	$SAT_{E2} - \frac{SAT_{E1} - SAT_{E2}}{SAT_a - 1}$	ConstService
SAT_B	B_{SAT}^q	$\frac{SAT_{E2}SAT_{SE2}\left(SAT_a - 1\right)^2\left(\text{Indicator}\left(SAT_a > 0\right) + \text{Indicator}\left(SAT_a < 0\right)\right)}{\left(SAT_{E1} - SAT_{E2}\right)^2}$	ConstService
Se0	S_{e0}	$SAT_B\left(SAT_A - vf_0\right)^2 \text{Indicator}\left(vf_0 > SAT_A\right)$	ConstService
vfe0	V_{FE0}	$KEvf_0 + Se_0$	ConstService
vref0	V_{ref0}	$vref$	PostInitService
VRU	$V_T V_{RMAX}$	$VRMAXcv$	VarService
VRL	$V_T V_{RMIN}$	$VRMINv$	VarService

Discretes

Name	Symbol	Type	Info
LL_LT	LT_{LL}	LessThan	
HG_lt	$None_{HG}$	LessThan	
LA_lim	lim_{LA}	AntiWindup	Limiter in Lag
SL	SL	LessThan	

Blocks

Name	Symbol	Type	Info
LG	LG	Lag	Transducer delay
SAT	SAT	ExcQuadSat	Field voltage saturation
LL	LL	LeadLag	Lead-lag compensator
HG	HG	HVGate	HVGate for under excitation
LA	LA	LagAntiWindup	Anti-windup lag
INT	INT	Integrator	Integrator
WF	WF	Washout	Feedback to input

Setpoints

Name	Target	Info
vref	vref0	Const reference voltage

Config Fields in [ESDC2A]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

EXST1

EXST1-type static excitation system.

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
syn		Synchronous generator idx			mandatory
TR	T_R	Measurement delay	0.010		
VIMAX	V_{IMAX}	Max. input voltage	0.200		
VIMIN	V_{IMIN}	Min. input voltage	0		
TC	T_C	LL numerator	1		
TB	T_B	LL denominator	1		
KA	K_A	Regulator gain	80		
TA	T_A	Regulator delay	0.050		
VRMAX	V_{RMAX}	Max. regulator output	8		
VRMIN	V_{RMIN}	Min. regulator output	-3		
KC	K_C	Coef. for Ifd	0.200		
KF	K_F	Feedback gain	0.100		
TF	T_F	Feedback delay	1		non_zero,non_negative
ue	u_e	effective online status	0		
Sn	S_m	Rated power from generator	0	*MVA*	
Vn	V_m	Rated voltage from generator	0	kV	
bus	bus	Bus idx of the generators	0		

Variables

Name	Symbol	Type	Description	Unit	Properties
LG_y	LG_y	State	State in lag transfer function		v_str
LL_x	LL_x	State	State in lead-lag		v_str
LR_y	LR_y	State	State in lag transfer function		v_str
WF_x	WF_x	State	State in washout filter		v_str
omega	ω	ExtState	Generator speed		
v	v	Algeb	Input to exciter (bus v or Eterm)		v_str
vout	$vout$	Algeb	Exciter final output voltage		v_str
vref	$vref$	Algeb	Reference voltage input	*p.u.*	v_str
vi	vi	Algeb	Total input voltages	*p.u.*	v_str
vl	vl	Algeb	Input after limiter		v_str
LL_y	LL_y	Algeb	Output of lead-lag		v_str
WF_y	WF_y	Algeb	Output of washout filter		v_str
vfmax	$vfmax$	Algeb	Upper bound of output limiter		v_str
vfmin	$vfmin$	Algeb	Lower bound of output limiter		v_str
vf	vf	ExtAlgeb	Excitation field voltage to generator		
XadIfd	$XadIfd$	ExtAlgeb	Armature excitation current		
a	a	ExtAlgeb	Bus voltage phase angle		
vbus	$vbus$	ExtAlgeb	Bus voltage magnitude		

Initialization Equations

Name	Symbol	Type	Initial Value
LG_y	LG_y	State	v
LL_x	LL_x	State	vl
LR_y	LR_y	State	$KALL_y$
WF_x	WF_x	State	LR_y
omega	ω	ExtState	
v	v	Algeb	$vbus$
vout	$vout$	Algeb	$uevf_0$
vref	$vref$	Algeb	$v + \frac{vf_0}{KA}$
vi	vi	Algeb	$\frac{vf_0}{KA}$
vl	vl	Algeb	$HLI_{zi}vi + HLI_{zl}VIMIN + HLI_{zu}VIMAX$
LL_y	LL_y	Algeb	vl
WF_y	WF_y	Algeb	0
vfmax	$vfmax$	Algeb	$-KCXadIfd + VRMAX$
vfmin	$vfmin$	Algeb	$-KCXadIfd + VRMIN$
vf	vf	ExtAlgeb	
XadIfd	$XadIfd$	ExtAlgeb	
a	a	ExtAlgeb	
vbus	$vbus$	ExtAlgeb	

Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
LG_y	LG_y	State	$-LG_y + v$	T_R
LL_x	LL_x	State	$-LL_x + vl$	T_B
LR_y	LR_y	State	$KALL_y - LR_y$	T_A
WF_x	WF_x	State	$LR_y - WF_x$	T_F
omega	ω	ExtState	0	

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
v	v	Algeb	$-v + vbus$
vout	$vout$	Algeb	$ue\left(HLR_{zi}LR_y + HLR_{zl}vfmin + HLR_{zu}vfmax\right) - vout$
vref	$vref$	Algeb	$-vref + vref_0$
vi	vi	Algeb	$-LG_y - WF_y - vi + vref$
vl	vl	Algeb	$HLI_{zi}vi + HLI_{zl}VIMIN + HLI_{zu}VIMAX - vl$
LL_y	LL_y	Algeb	$LL_{LTz1}\left(LL_y - vl\right) + LL_xTB - LL_yTB + TC\left(-LL_x + vl\right)$
WF_y	WF_y	Algeb	$KF\left(LR_y - WF_x\right) - TFWF_y$
vfmax	$vfmax$	Algeb	$-KCXadIfd + VRMAX - vfmax$
vfmin	$vfmin$	Algeb	$-KCXadIfd + VRMIN - vfmin$
vf	vf	ExtAlgeb	$ue\left(-vf_0 + vout\right)$
XadIfd	$XadIfd$	ExtAlgeb	0
a	a	ExtAlgeb	0
vbus	$vbus$	ExtAlgeb	0

Services

Name	Symbol	Equation	Type
vref0	V_{ref0}	$vref$	PostInitService

Discretes

Name	Symbol	Type	Info
HLI	HLI	HardLimiter	Hard limiter on input
LL_LT	LT_{LL}	LessThan	
HLR	HLR	HardLimiter	Hard limiter on regulator output

Blocks

Name	Symbol	Type	Info
LG	LG	Lag	Sensing delay
LL	LL	LeadLag	Lead-lag compensator
LR	LR	Lag	Regulator
WF	WF	Washout	Stablizing circuit feedback

Setpoints

Name	Target	Info
vref	vref0	constant vref

Config Fields in [EXST1]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

ESST3A

Static exciter type 3A model

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
syn		Synchronous generator idx			mandatory
TR	T_R	Sensing time constant	0.010	*p.u.*	
VIMAX	V_{IMAX}	Max. input voltage	0.800		
VIMIN	V_{IMIN}	Min. input voltage	-0.100		
KM	K_M	Forward gain constant	500		
TC	T_C	Lead time constant in lead-lag	3		
TB	T_B	Lag time constant in lead-lag	15		
KA	K_A	Gain in anti-windup lag TF	50		
TA	T_A	Lag time constant in anti-windup lag	0.100		
VRMAX	V_{RMAX}	Maximum excitation limit	8	*p.u.*	
VRMIN	V_{RMIN}	Minimum excitation limit	0	*p.u.*	
KG	K_G	Feedback gain of inner field regulator	1		
KP	K_P	Potential circuit gain coeff.	4		
KI	K_I	Potential circuit gain coeff.	0.100		
VBMAX	V_{BMAX}	VB upper limit	18	*p.u.*	
KC	K_C	Rectifier loading factor proportional to commutating reactance	0.100		
XL	X_L	Potential source reactance	0.010		
VGMAX	V_{GMAX}	VG upper limit	4	*p.u.*	
THETAP	θ_P	Rectifier firing angle	0	*degree*	
TM	K_C	Inner field regulator forward time constant	0.100		
VMMAX	V_{MMAX}	Maximum VM limit	1	*p.u.*	
VMMIN	V_{RMIN}	Minimum VM limit	0.100	*p.u.*	
ue	u_e	effective online status	0		
Sn	S_m	Rated power from generator	0	*MVA*	
Vn	V_m	Rated voltage from generator	0	*kV*	
bus	bus	Bus idx of the generators	0		

Variables

Name	Symbol	Type	Description	Unit	Properties
LG_y	LG_y	State	State in lag transfer function		v_str
LL_x	LL_x	State	State in lead-lag		v_str
LAW1_y	LAW_{1y}	State	State in lag TF		v_str
LAW2_y	LAW_{2y}	State	State in lag TF		v_str
omega	ω	ExtState	Generator speed		
v	v	Algeb	Input to exciter (bus v or Eterm)		v_str
vout	$vout$	Algeb	Exciter final output voltage		v_str
UEL	UEL	Algeb	Interface var for under exc. limiter		v_str
IN	IN	Algeb	Input to FEX		v_str
FEX_y	FEX_y	Algeb	Output of piecewise		v_str
VB_x	VB_x	Algeb	Value before limiter		v_str
VG_x	VG_x	Algeb	Value before limiter		v_str
vrs	vrs	Algeb	VR subtract feedback VG		v_str
vref	$vref$	Algeb	Reference voltage input	*p.u.*	v_str
vi	vi	Algeb	Total input voltages	*p.u.*	v_str
vil	vil	Algeb	Input voltage after limit		v_str
LL_y	LL_y	Algeb	Output of lead-lag		v_str
vf	vf	ExtAlgeb	Excitation field voltage to generator		
XadIfd	$XadIfd$	ExtAlgeb	Armature excitation current		
a	a	ExtAlgeb	Bus voltage phase angle		
vbus	$vbus$	ExtAlgeb	Bus voltage magnitude		
vd	vd	ExtAlgeb	d-axis machine voltage		
vq	vq	ExtAlgeb	q-axis machine voltage		
Id	Id	ExtAlgeb	d-axis machine current		
Iq	Iq	ExtAlgeb	q-axis machine current		

Initialization Equations

Name	Symbol	Type	Initial Value
LG_y	LG_y	State	v
LL_x	LL_x	State	$HG_{ltz0}UEL + HG_{ltz1}vil$
LAW1	LAW_1	State	$KALL_y$
LAW2	LAW_2	State	$KMvrs$
omega	ω	ExtStat	
v	v	Algeb	$vbus$
vout	$vout$	Algeb	$uevf_0$
UEL	UEL	Algeb	UEL_0
IN	IN	Algeb	$\text{safe}_{div}\left(KCXadIfd, VE\right)$
FEX_y	FEX_y	Algeb	$\text{FixPiecewise}\left(\left(1,\ IN \leq 0\right), \left(1 - 0.577IN,\ IN \leq 0.433\right), \left(\sqrt{0.75 - IN^2},\ IN \leq 0.75\right), (1.73\right.$
VB_x	VB_x	Algeb	FEX_yVE
VG_x	VG_x	Algeb	$KGvout$
vrs	vrs	Algeb	$\frac{\text{safe}_{div}\left(vf_0, VBMAXVB_{limzu} + VB_{limzi}VB_x\right)}{KM}$
vref	$vref$	Algeb	$v + \frac{VGMAXVG_{limzu} + VG_{limzi}VG_x + vrs}{KA}$
vi	vi	Algeb	$-v + vref$
vil	vil	Algeb	$HLI_{zi}vi + HLI_{zl}VIMIN + HLI_{zu}VIMAX$
LL_y	LL_y	Algeb	$HG_{ltz0}UEL + HG_{ltz1}vil$
vf	vf	ExtAlgeb	
XadIfd	$XadIfd$	ExtAlgeb	
a	a	ExtAlgeb	
vbus	$vbus$	ExtAlgeb	
vd	vd	ExtAlgeb	
vq	vq	ExtAlgeb	

Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
LG_y	LG_y	State	$-LG_y + v$	T_R
LL_x	LL_x	State	$HG_{ltz0}UEL + HG_{ltz1}vil - LL_x$	T_B
LAW1_y	LAW_{1y}	State	$KALL_y - LAW_{1y}$	T_A
LAW2_y	LAW_{2y}	State	$KMvrs - LAW_{2y}$	K_C
omega	ω	ExtState	0	

Algebraic Equations

Nam	Sym-bol	Type	RHS of Equation "0 = g(x, y)"
v	v	Al-geb	$-v + vbus$
vout	$vout$	Al-geb	$LAW_{2y} ue \left(VBMAXVB_{limzu} + VB_{limzi} VB_x\right) - vout$
UEL	UEL	Al-geb	$-UEL + UEL_0$
IN	IN	Al-geb	$ue \left(-INVE + KCXadIfd\right)$
FEX_	$FEX_$	Al-geb	$-FEX_y$+FixPiecewise $\Big(\left(1,\ IN \leq 0\right), \left(1 - 0.577IN,\ IN \leq 0.433\right), \Big(\sqrt{0.75 - IN^2},\ IN \leq 0.75$
VB_x	VB_x	Al-geb	$FEX_y VE - VB_x$
VG_x	VG_x	Al-geb	$KGvout - VG_x$
vrs	vrs	Al-geb	$LAW_{1y} - VGMAXVG_{limzu} - VG_{limzi} VG_x - vrs$
vref	$vref$	Al-geb	$-vref + vref_0$
vi	vi	Al-geb	$-LG_y - vi + vref$
vil	vil	Al-geb	$HLI_{zi} vi + HLI_{zl} VIMIN + HLI_{zu} VIMAX - vil$
LL_y	LL_y	Al-geb	$LL_{LTz1} \left(-HG_{ltz0} UEL - HG_{ltz1} vil + LL_y\right) + LL_x TB - LL_y TB + TC \left(HG_{ltz0} UEL + HG_{ltz1} vil - LL_x\right)$
vf	vf	Ex-tAl-geb	$ue \left(-vf_0 + vout\right)$
Xad-Ifd	$XadI$	Ex-tAl-geb	0
a	a	Ex-tAl-geb	0
vbus	$vbus$	Ex-tAl-geb	0
vd	vd	Ex-tAl-geb	0
vq	vq	Ex-tAl-geb	0
Id	Id	Ex-tAl-geb	0
Iq	Iq	Ex-tAl-	0

Services

Name	Symbol	Equation	Type
KPC	K_{PC}	$KPe^{i\,\text{radians}\,(THETAP)}$	ConstService
UEL0	U_{EL0}	-9999	ConstService
VE	V_E	$\lvert KPC\,(vd + ivq) + i\,(Id + iIq)\,(KI + KPCXL)\rvert$	VarService
vref0	V_{ref0}	$vref$	PostInitService

Discretes

Name	Symbol	Type	Info
VB_lim	lim_{VB}	HardLimiter	
VG_lim	lim_{VG}	HardLimiter	
HG_lt	$None_{HG}$	LessThan	
LL_LT	LT_{LL}	LessThan	
LAW1_lim	lim_{LAW1}	AntiWindup	Limiter in Lag
HLI	HLI	HardLimiter	Input limiter
LAW2_lim	lim_{LAW2}	AntiWindup	Limiter in Lag

Blocks

Name	Symbol	Type	Info
LG	LG	Lag	Voltage transducer
FEX	FEX	Piecewise	Piecewise function FEX
VB	VB	GainLimiter	VB with limiter
VG	VG	GainLimiter	Feedback gain with HL
HG	HG	HVGate	HVGate for under excitation
LL	LL	LeadLag	Regulator
LAW1	$LAW1$	LagAntiWindup	Lag AW on VR
LAW2	$LAW2$	LagAntiWindup	Lag AW on VM

Setpoints

Name	Target	Info
vref	vref0	Initial reference voltage input

Config Fields in [ESST3A]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

SEXS

Simplified Excitation System

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
syn		Synchronous generator idx			mandatory
TATB	T_A/T_B	Time constant TA/TB	0.400		
TB	T_B	Time constant TB in LL	5		
K	K	Gain	20		non_zero
TE	T_E	AW Lag time constant	1		
EMIN	E_{MIN}	lower limit	-99		
EMAX	E_{MAX}	upper limit	99		
ue	u_e	effective online status	0		
Sn	S_m	Rated power from generator	0	*MVA*	
Vn	V_m	Rated voltage from generator	0	*kV*	
bus	bus	Bus idx of the generators	0		

Variables

Name	Symbol	Type	Description	Unit	Properties
LL_x	LL_x	State	State in lead-lag		v_str
LAW_y	LAW_y	State	State in lag TF		v_str
omega	ω	ExtState	Generator speed		
v	v	Algeb	Input to exciter (bus v or Eterm)		v_str
vout	$vout$	Algeb	Exciter final output voltage		v_str
vref	$vref$	Algeb	Reference voltage input	*p.u.*	v_str
vi	vi	Algeb	Total input voltages	*p.u.*	v_str
LL_y	LL_y	Algeb	Output of lead-lag		v_str
vf	vf	ExtAlgeb	Excitation field voltage to generator		
XadIfd	$XadIfd$	ExtAlgeb	Armature excitation current		
a	a	ExtAlgeb	Bus voltage phase angle		
vbus	$vbus$	ExtAlgeb	Bus voltage magnitude		

Initialization Equations

Name	Symbol	Type	Initial Value
LL_x	LL_x	State	vi
LAW_y	LAW_y	State	KLL_y
omega	ω	ExtState	
v	v	Algeb	$vbus$
vout	$vout$	Algeb	$uevf_0$
vref	$vref$	Algeb	$v + \frac{vf_0}{K}$
vi	vi	Algeb	$\frac{vf_0}{K}$
LL_y	LL_y	Algeb	vi
vf	vf	ExtAlgeb	
XadIfd	$XadIfd$	ExtAlgeb	
a	a	ExtAlgeb	
vbus	$vbus$	ExtAlgeb	

Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
LL_x	LL_x	State	$-LL_x + vi$	T_B
LAW_y	LAW_y	State	$KLL_y - LAW_y$	T_E
omega	ω	ExtState	0	

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
v	v	Algeb	$-v + vbus$
vout	$vout$	Algeb	$LAW_y ue - vout$
vref	$vref$	Algeb	$-vref + vref_0$
vi	vi	Algeb	$-v - vi + vref$
LL_y	LL_y	Algeb	$LL_{LTz1} (LL_y - vi) + LL_x TB - LL_y TB + TA (-LL_x + vi)$
vf	vf	ExtAlgeb	$ue (-vf_0 + vout)$
XadIfd	$XadIfd$	ExtAlgeb	0
a	a	ExtAlgeb	0
vbus	$vbus$	ExtAlgeb	0

Services

Name	Symbol	Equation	Type
TA	TA	$TATBTB$	ConstService
vref0	V_{ref0}	$vref$	PostInitService

Discretes

Name	Symbol	Type	Info
LL_LT	LT_{LL}	LessThan	
LAW_lim	lim_{LAW}	AntiWindup	Limiter in Lag

Blocks

Name	Symbol	Type	Info
LL	LL	LeadLag	
LAW	LAW	LagAntiWindup	

Setpoints

Name	Target	Info
vref	vref0	Constant vref

Config Fields in [SEXS]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

IEEET1

IEEET1 exciter.

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
syn		Synchronous generator idx			mandatory
TR	T_R	Sensing time constant	0.020	*p.u.*	
KA	K_A	Regulator gain	5	*p.u.*	
TA	T_A	Lag time constant in anti-windup lag	0.040	*p.u.*	
VRMAX	V_{RMAX}	Maximum excitation limit	7.300	*p.u.*	
VRMIN	V_{RMIN}	Minimum excitation limit	-7.300	*p.u.*	
KE	K_E	Gain added to saturation	1	*p.u.*	
TE	T_E	Exciter integrator time constant	0.800	*p.u.*	
KF	K_F	Feedback gain	0.100		
TF	T_F	Feedback delay	1		non_zero,non_negative
Switch	S_w	Switch unused in PSS/E	0	*bool*	
E1	E_1	First saturation point	0	*p.u.*	
SE1	S_{E1}	Value at first saturation point	0	*p.u.*	
E2	E_2	Second saturation point	1	*p.u.*	
SE2	S_{E2}	Value at second saturation point	1	*p.u.*	
ue	u_e	effective online status	0		
Sn	S_m	Rated power from generator	0	*MVA*	
Vn	V_m	Rated voltage from generator	0	*kV*	
bus	bus	Bus idx of the generators	0		

Variables

Name	Symbol	Type	Description	Unit	Properties
LG_y	LG_y	State	State in lag transfer function		v_str
LA_y	LA_y	State	State in lag TF		v_str
INT_y	INT_y	State	Integrator output		v_str
WF_x	WF_x	State	State in washout filter		v_str
omega	ω	ExtState	Generator speed		
v	v	Algeb	Input to exciter (bus v or Eterm)		v_str
vout	$vout$	Algeb	Exciter final output voltage		v_str
vref	$vref$	Algeb	Reference voltage input	*p.u.*	v_str
vi	vi	Algeb	Total input voltages	*p.u.*	v_str
VFE	VFE	Algeb	Combined saturation feedback	*p.u.*	v_str
Se	Se	Algeb	saturation output		v_str
WF_y	WF_y	Algeb	Output of washout filter		v_str
vf	vf	ExtAlgeb	Excitation field voltage to generator		
XadIfd	$XadIfd$	ExtAlgeb	Armature excitation current		
a	a	ExtAlgeb	Bus voltage phase angle		
vbus	$vbus$	ExtAlgeb	Bus voltage magnitude		

Initialization Equations

Name	Symbol	Type	Initial Value
LG_y	LG_y	State	v
LA_y	LA_y	State	$KAue\left(-WF_y + vi\right)$
INT_y	INT_y	State	vf_0
WF_x	WF_x	State	$vout$
omega	ω	ExtState	
v	v	Algeb	$vbus$
vout	$vout$	Algeb	$uevf_0$
vref	$vref$	Algeb	$v + vb_0$
vi	vi	Algeb	$-v + vref$
VFE	VFE	Algeb	vfe_0
Se	Se	Algeb	Se_0
WF_y	WF_y	Algeb	0
vf	vf	ExtAlgeb	
XadIfd	$XadIfd$	ExtAlgeb	
a	a	ExtAlgeb	
vbus	$vbus$	ExtAlgeb	

Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
LG_y	LG_y	State	$-LG_y + v$	T_R
LA_y	LA_y	State	$KAue\left(-WF_y + vi\right) - LA_y$	T_A
INT_y	INT_y	State	$ue\left(LA_y - VFE\right)$	T_E
WF_x	WF_x	State	$-WF_x + vout$	T_F
omega	ω	ExtState	0	

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
v	v	Algeb	$-v + vbus$
vout	$vout$	Algeb	$ue\left(INT_y - vout\right)$
vref	$vref$	Algeb	$-vref + vref_0$
vi	vi	Algeb	$ue\left(-LG_y - vi + vref\right)$
VFE	VFE	Algeb	$ue\left(INT_yKE + Se - VFE\right)$
Se	Se	Algeb	$SAT_BSL_{z0}\left(INT_y - SAT_A\right)^2 - Se$
WF_y	WF_y	Algeb	$KF\left(-WF_x + vout\right) - TFWF_y$
vf	vf	ExtAlgeb	$ue\left(-vf_0 + vout\right)$
XadIfd	$XadIfd$	ExtAlgeb	0
a	a	ExtAlgeb	0
vbus	$vbus$	ExtAlgeb	0

Services

Name	Symbol	Equation	Type
VRMAXc	$VRMAXc$	$VRMAX - 999_z VRM + 999$	ConstService
SAT_E1	E_{SAT}^{1c}	E_1	ConstService
SAT_E2	E_{SAT}^{2c}	$E_2 - 2SAT_{zSE2} + 2$	ConstService
SAT_SE1	SE_{SAT}^{1c}	SE_1	ConstService
SAT_SE2	SE_{SAT}^{2c}	$-2SAT_{zSE2} + SE_2 + 2$	ConstService
SAT_a	a_{SAT}	$\sqrt{\frac{SAT_{E1}SAT_{SE1}}{SAT_{E2}SAT_{SE2}}}$ (Indicator $(SAT_{SE2} > 0)$ + Indicator $(SAT_{SE2}$	ConstService
SAT_A	A_{SAT}^{q}	$SAT_{E2} - \frac{SAT_{E1} - SAT_{E2}}{SAT_a - 1}$	ConstService
SAT_B	B_{SAT}^{q}	$\frac{SAT_{E2}SAT_{SE2}(SAT_a - 1)^2(\text{Indicator}\,(SAT_a > 0) + \text{Indicator}\,(SAT_a < 0))}{(SAT_{E1} - SAT_{E2})^2}$	ConstService
Se0	S_{e0}	$SAT_B\,(SAT_A - vf_0)^2$ Indicator $(vf_0 > SAT_A)$	ConstService
vr0	V_{r0}	$KEvf_0 + Se_0$	ConstService
vb0	V_{b0}	$\frac{vr_0}{KA}$	ConstService
vfe0	V_{FE0}	$KEvf_0 + Se_0$	ConstService
vref0	V_{ref0}	$vref$	PostInitService

Discretes

Name	Symbol	Type	Info
LA_lim	lim_{LA}	AntiWindup	Limiter in Lag
SL	SL	LessThan	

Blocks

Name	Symbol	Type	Info
SAT	SAT	ExcQuadSat	Field voltage saturation
LG	LG	Lag	Sensing delay
LA	LA	LagAntiWindup	Anti-windup lag
INT	INT	Integrator	Integrator
WF	WF	Washout	Stablizing circuit feedback

Setpoints

Name	Target	Info
vref	vref0	Const reference voltage

Config Fields in [IEEET1]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

EXAC1

EXAC1 model.

Ref: https://www.powerworld.com/WebHelp/Content/TransientModels_HTML/Exciter%20EXAC1.htm

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
syn		Synchronous generator idx			mandatory
TR	T_R	Sensing time constant	0.010	*p.u.*	
TB	T_B	Lag time constant in lead-lag	1	*p.u.*	
TC	T_C	Lead time constant in lead-lag	1	*p.u.*	
KA	K_A	Regulator gain	80		
TA	T_A	Lag time constant in regulator	0.040	*p.u.*	
VRMAX	V_{RMAX}	Maximum regulator output	8	*p.u.*	
VRMIN	V_{RMIN}	Minimum regulator output	0	*p.u.*	
TE	T_E	Exciter integrator time constant	0.800	*p.u.*	non_negative
KF	K_F	Feedback gain	0.100		
TF	T_F	Feedback delay	1		non_zero,non_negative
KC	K_C	Rectifier loading factor proportional to commutating reactance	0.100		
KD	K_C	Ifd feedback gain	0		
KE	K_E	Saturation feedback gain	1	*p.u.*	
E1	E_1	First saturation point	0	*p.u.*	
SE1	S_{E1}	Value at first saturation point	0	*p.u.*	
E2	E_2	Second saturation point	1	*p.u.*	
SE2	S_{E2}	Value at second saturation point	1	*p.u.*	
ue	u_e	effective online status	0		
Sn	S_m	Rated power from generator	0	*MVA*	
Vn	V_m	Rated voltage from generator	0	*kV*	
bus	bus	Bus idx of the generators	0		

Variables

Name	Symbol	Type	Description	Unit	Properties
LG_y	LG_y	State	State in lag transfer function		v_str
LL_x	LL_x	State	State in lead-lag		v_str
LA_y	LA_y	State	State in lag TF		v_str
INT_y	INT_y	State	Integrator output		v_str,v_iter
WF_x	WF_x	State	State in washout filter		v_str
omega	ω	ExtState	Generator speed		
v	v	Algeb	Input to exciter (bus v or Eterm)		v_str
vout	$vout$	Algeb	Exciter final output voltage		v_str
IN	IN	Algeb	Input to FEX		v_str,v_iter
FEX_y	FEX_y	Algeb	Output of piecewise		v_str,v_iter
vi	vi	Algeb	Total input voltages	*p.u.*	v_str
LL_y	LL_y	Algeb	Output of lead-lag		v_str
Se	Se	Algeb	saturation output		v_str
VFE	VFE	Algeb	Combined saturation feedback	*p.u.*	v_str
vref	$vref$	Algeb	Reference voltage input	*p.u.*	v_str
WF_y	WF_y	Algeb	Output of washout filter		v_str
vf	vf	ExtAlgeb	Excitation field voltage to generator		
XadIfd	$XadIfd$	ExtAlgeb	Armature excitation current		
a	a	ExtAlgeb	Bus voltage phase angle		
vbus	$vbus$	ExtAlgeb	Bus voltage magnitude		

Initialization Equations

Nam	Sym-bol	Type	Initial Value
LG_y	LG_y	State	v
LL_x	LL_x	State	vi
LA_y	LA_y	State	$KALL_y$
INT_	INT_y	State	$FEX_yINT_y - vf_0$
WF_	WF_x	State	VFE
omeg	ω	ExtSta	
v	v	Al-geb	$vbus$
vout	$vout$	Al-geb	$uevf_0$
IN	IN	Al-geb	$-ININT_y + KCXadIfd$
FEX_	$FEX_$	Al-geb	$-FEX_y$+FixPiecewise $\left((1,\ IN \leq 0), (1 - 0.577IN,\ IN \leq 0.433), \left(\sqrt{0.75 - IN^2},\ IN \leq 0.7\right.\right.$
vi	vi	Al-geb	$-v + vref$
LL_y	LL_y	Al-geb	vi
Se	Se	Al-geb	$SAT_B\left(INT_y - SAT_A\right)^2$ Indicator $\left(INT_y > SAT_A\right)$
VFE	VFE	Al-geb	$INT_yKE + KDXadIfd + Se$
vref	$vref$	Al-geb	$v + \frac{VFE}{KA}$
WF_	WF_y	Al-geb	0
vf	vf	Ex-tAl-geb	
Xad-Ifd	$XadI$	Ex-tAl-geb	
a	a	Ex-tAl-geb	
vbus	$vbus$	Ex-tAl-geb	

Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
LG_y	LG_y	State	$-LG_y + v$	T_R
LL_x	LL_x	State	$-LL_x + vi$	T_B
LA_y	LA_y	State	$KALL_y - LA_y$	T_A
INT_y	INT_y	State	$ue\left(LA_y - VFE\right)$	T_E
WF_x	WF_x	State	$VFE - WF_x$	T_F
omega	ω	ExtState	0	

Algebraic Equations

Nam	Sym-bol	Type	RHS of Equation "0 = g(x, y)"
v	v	Al-geb	$-v + vbus$
vout	$vout$	Al-geb	$ue\left(FEX_y INT_y - vout\right)$
IN	IN	Al-geb	$ue\left(-ININT_y + KCXadIfd\right)$
FEX_	$FEX_{}$	Al-geb	$-FEX_y$+FixPiecewise $\Big(\left(1,\ IN \leq 0\right), \left(1 - 0.577IN,\ IN \leq 0.433\right), \Big(\sqrt{0.75 - IN^2},\ IN \leq 0.7$
vi	vi	Al-geb	$ue\left(-WF_y - v - vi + vref\right)$
LL_y	LL_y	Al-geb	$LL_{LTz1}\left(LL_y - vi\right) + LL_x TB - LL_y TB + TC\left(-LL_x + vi\right)$
Se	Se	Al-geb	$ue\left(SAT_B SL_{z0}\left(INT_y - SAT_A\right)^2 - Se\right)$
VFE	VFE	Al-geb	$ue\left(INT_y KE + KDXadIfd + Se - VFE\right)$
vref	$vref$	Al-geb	$-vref + vref_0$
WF_	WF_y	Al-geb	$KF\left(VFE - WF_x\right) - TFWF_y$
vf	vf	Ex-tAl-geb	$ue\left(-vf_0 + vout\right)$
Xad-Ifd	$XadI$	Ex-tAl-geb	0
a	a	Ex-tAl-geb	0
vbus	$vbus$	Ex-tAl-geb	0

Services

Name	Symbol	Equation	Type
SAT_E1	E_{SAT}^{1c}	E_1	ConstService
SAT_E2	E_{SAT}^{2c}	$E_2 - 2SAT_{zSE2} + 2$	ConstService
SAT_SE1	SE_{SAT}^{1c}	SE_1	ConstService
SAT_SE2	SE_{SAT}^{2c}	$-2SAT_{zSE2} + SE_2 + 2$	ConstService
SAT_a	a_{SAT}	$\sqrt{\frac{SAT_{E1}SAT_{SE1}}{SAT_{E2}SAT_{SE2}}}$ (Indicator $(SAT_{SE2} > 0)$ + Indicator $(SAT_{SE2} <$	ConstService
SAT_A	A_{SAT}^{q}	$SAT_{E2} - \frac{SAT_{E1}-SAT_{E2}}{SAT_a-1}$	ConstService
SAT_B	B_{SAT}^{q}	$\frac{SAT_{E2}SAT_{SE2}(SAT_a-1)^2(\text{Indicator }(SAT_a>0)+\text{Indicator }(SAT_a<0))}{(SAT_{E1}-SAT_{E2})^2}$	ConstService
vref0	V_{ref0}	$vref$	PostInitService

Discretes

Name	Symbol	Type	Info
LL_LT	LT_{LL}	LessThan	
LA_lim	lim_{LA}	AntiWindup	Limiter in Lag
SL	SL	LessThan	

Blocks

Name	Symbol	Type	Info
SAT	SAT	ExcQuadSat	Field voltage saturation
FEX	FEX	Piecewise	Piecewise function FEX
LG	LG	Lag	Voltage transducer
LL	LL	LeadLag	Regulator
LA	LA	LagAntiWindup	Lag AW on VR
INT	INT	Integrator	Integrator
WF	WF	Washout	Stablizing circuit feedback

Setpoints

Name	Target	Info
vref	vref0	Initial reference voltage input

Config Fields in [EXAC1]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

EXAC2

EXAC2 model.

Ref: https://www.powerworld.com/WebHelp/Content/TransientModels_HTML/Exciter%20EXAC2.htm

Notes

`VLR` is an input parameter, but to initialize the LVGate, an internal `VLRx` will be computed as a contant upon initialization. The constant `VLRx` will be used in the place of `VLR` in the block diagram.

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
syn		Synchronous generator idx			mandatory
TR	T_R	Sensing time constant	0.010	*p.u.*	
TB	T_B	Lag time constant in lead-lag	1	*p.u.*	
TC	T_C	Lead time constant in lead-lag	1	*p.u.*	
KA	K_A	Regulator gain	80		
TA	T_A	Lag time constant in regulator	0.040	*p.u.*	
VRMAX	V_{RMAX}	Maximum regulator output	8	*p.u.*	
VRMIN	V_{RMIN}	Minimum regulator output	0	*p.u.*	
TE	T_E	Exciter integrator time constant	0.800	*p.u.*	non_negative
KF	K_F	Feedback gain	0.100		

continues on next

Table 8 – continued from previous page

Name	Symbol	Description	Default	Unit	Properties
TF	T_F	Feedback delay	1		non_zero,non_neg
KC	K_C	Rectifier loading factor proportional to commutating reactance	0.100		
KD	K_C	Ifd feedback gain	0		
KE	K_E	Saturation feedback gain	1	*p.u.*	
E1	E_1	First saturation point	0	*p.u.*	
SE1	S_{E1}	Value at first saturation point	0	*p.u.*	
E2	E_2	Second saturation point	1	*p.u.*	
SE2	S_{E2}	Value at second saturation point	1	*p.u.*	
VAMAX	V_{RMAX}	Maximum KA block output	8	*p.u.*	
VAMIN	V_{RMIN}	Minimum KA block output	0	*p.u.*	
VLR	V_{LR}	low voltage constant	0	*p.u.*	
KL	K_L	gain for low voltage	1	*p.u.*	
KH	K_H	gain for high voltage	1	*p.u.*	
KB	K_B	gain KB for regulator	1	*p.u.*	non_negative
ue	u_e	effective online status	0		
Sn	S_m	Rated power from generator	0	*MVA*	
Vn	V_m	Rated voltage from generator	0	*kV*	
bus	bus	Bus idx of the generators	0		

Variables

Name	Symbol	Type	Description	Unit	Properties
LG_y	LG_y	State	State in lag transfer function		v_str
LL_x	LL_x	State	State in lead-lag		v_str
LA_y	LA_y	State	State in lag TF		v_str
INT_y	INT_y	State	Integrator output		v_str,v_iter
WF_x	WF_x	State	State in washout filter		v_str
omega	ω	ExtState	Generator speed		
v	v	Algeb	Input to exciter (bus v or Eterm)		v_str
vout	$vout$	Algeb	Exciter final output voltage		v_str
IN	IN	Algeb	Input to FEX		v_str,v_iter
FEX_y	FEX_y	Algeb	Output of piecewise		v_str,v_iter
vi	vi	Algeb	Total input voltages	*p.u.*	v_str
LL_y	LL_y	Algeb	Output of lead-lag		v_str
Se	Se	Algeb	saturation output		v_str
VFE	VFE	Algeb	Combined saturation feedback	*p.u.*	v_str
vref	$vref$	Algeb	Reference voltage input	*p.u.*	v_str
WF_y	WF_y	Algeb	Output of washout filter		v_str
VHA	VHA	Algeb			v_str
VL	VL	Algeb			v_str
VR_x	VR_x	Algeb	Value before limiter		v_str
VLRx	$VLRx$	Algeb			v_str
vf	vf	ExtAlgeb	Excitation field voltage to generator		
XadIfd	$XadIfd$	ExtAlgeb	Armature excitation current		
a	a	ExtAlgeb	Bus voltage phase angle		
vbus	$vbus$	ExtAlgeb	Bus voltage magnitude		

Initialization Equations

Nam	Sym-bol	Type	Initial Value
LG_y	LG_y	State	v
LL_x	LL_x	State	vi
LA_y	LA_y	State	$KALL_y$
INT_	INT_y	State	$FEX_yINT_y - vf_0$
WF_	WF_x	State	VFE
omeg	ω	ExtSta	
v	v	Al-geb	$vbus$
vout	$vout$	Al-geb	$uevf_0$
IN	IN	Al-geb	$-ININT_y + KCXadIfd$
FEX_	$FEX_$	Al-geb	$-FEX_y$+FixPiecewise $\Big((1,\ IN \leq 0), (1 - 0.577IN,\ IN \leq 0.433), \Big(\sqrt{0.75 - IN^2},\ IN \leq 0.75$
vi	vi	Al-geb	$-v + vref$
LL_y	LL_y	Al-geb	vi
Se	Se	Al-geb	$SAT_B\left(INT_y - SAT_A\right)^2$ Indicator $\left(INT_y > SAT_A\right)$
VFE	VFE	Al-geb	$INT_yKE + KDXadIfd + Se$
vref	$vref$	Al-geb	$v + \frac{KLVFE + \frac{VFE}{KB}}{KA}$
WF_	WF_y	Al-geb	0
VHA	VHA	Al-geb	$-KHVFE + LA_y$
VL	VL	Al-geb	$KL\left(-VFE + VLRx\right)$
VR_	VR_x	Al-geb	$KB\left(LVG_{ltz0}VL + LVG_{ltz1}VHA\right)$
VLR	VLR	Al-geb	VLR Indicator $\left(VFE + \frac{VFE}{KBKL} < VLR\right) + \left(VFE + \frac{VFE}{KBKL}\right)$ Indicator $\left(VFE + \frac{VFE}{KBKL} > VLR\right)$
vf	vf	Ex-tAl-geb	
Xad-Ifd	$XadI$	Ex-tAl-geb	
a	a	Ex-tAl-geb	
vbus	$vbus$	Ex-tAl-geb	

Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
LG_y	LG_y	State	$-LG_y + v$	T_R
LL_x	LL_x	State	$-LL_x + vi$	T_B
LA_y	LA_y	State	$KALL_y - LA_y$	T_A
INT_y	INT_y	State	$ue\left(-VFE + VRMAXVR_{limzu} + VRMINVR_{limzl} + VR_{lim}\right.$	T_E
WF_x	WF_x	State	$VFE - WF_x$	T_F
omega	ω	ExtState	0	

Algebraic Equations

Nam	Sym-bol	Type	RHS of Equation "0 = g(x, y)"
v	v	Al-geb	$-v + vbus$
vout	$vout$	Al-geb	$ue\left(FEX_y INT_y - vout\right)$
IN	IN	Al-geb	$ue\left(-ININT_y + KCXadIfd\right)$
FEX_	$FEX_$	Al-geb	$-FEX_y$+FixPiecewise $\Big(\left(1,\ IN \leq 0\right), \left(1 - 0.577IN,\ IN \leq 0.433\right), \Big(\sqrt{0.75 - IN^2},\ IN \leq 0.7$
vi	vi	Al-geb	$ue\left(-WF_y - v - vi + vref\right)$
LL_y	LL_y	Al-geb	$LL_{LTz1}\left(LL_y - vi\right) + LL_x TB - LL_y TB + TC\left(-LL_x + vi\right)$
Se	Se	Al-geb	$ue\left(SAT_B SL_{z0}\left(INT_y - SAT_A\right)^2 - Se\right)$
VFE	VFE	Al-geb	$ue\left(INT_y KE + KDXadIfd + Se - VFE\right)$
vref	$vref$	Al-geb	$-vref + vref_0$
WF_	WF_y	Al-geb	$KF\left(VFE - WF_x\right) - TFWF_y$
VHA	VHA	Al-geb	$-KHVFE + LA_y - VHA$
VL	VL	Al-geb	$KL\left(-VFE + VLRx\right) - VL$
VR_	VR_x	Al-geb	$KB\left(LVG_{ltz0} VL + LVG_{ltz1} VHA\right) - VR_x$
VLR	VLR	Al-geb	$VLR_0 - VLRx$
vf	vf	Ex-tAl-geb	$ue\left(-vf_0 + vout\right)$
Xad-Ifd	$XadI$	Ex-tAl-geb	0
a	a	Ex-tAl-geb	0
vbus	$vbus$	Ex-tAl-geb	0

Services

Name	Symbol	Equation	Type
SAT_E1	E_{SAT}^{1c}	E_1	ConstService
SAT_E2	E_{SAT}^{2c}	$E_2 - 2SAT_{zSE2} + 2$	ConstService
SAT_SE1	SE_{SAT}^{1c}	SE_1	ConstService
SAT_SE2	SE_{SAT}^{2c}	$-2SAT_{zSE2} + SE_2 + 2$	ConstService
SAT_a	a_{SAT}	$\sqrt{\frac{SAT_{E1}SAT_{SE1}}{SAT_{E2}SAT_{SE2}}}$ (Indicator $(SAT_{SE2} > 0)$ + Indicator $(SAT_{SE2} <$	ConstService
SAT_A	A_{SAT}^{q}	$SAT_{E2} - \frac{SAT_{E1}-SAT_{E2}}{SAT_a-1}$	ConstService
SAT_B	B_{SAT}^{q}	$\frac{SAT_{E2}SAT_{SE2}(SAT_a-1)^2(\text{Indicator }(SAT_a>0)+\text{Indicator }(SAT_a<0))}{(SAT_{E1}-SAT_{E2})^2}$	ConstService
vref0	V_{ref0}	$vref$	PostInitService
VLR0	$VLR0$	$VLRx$	PostInitService

Discretes

Name	Symbol	Type	Info
LL_LT	LT_{LL}	LessThan	
LA_lim	lim_{LA}	AntiWindup	Limiter in Lag
SL	SL	LessThan	
LVG_lt	$None_{LVG}$	LessThan	
VR_lim	lim_{VR}	HardLimiter	

Blocks

Name	Symbol	Type	Info
SAT	SAT	ExcQuadSat	Field voltage saturation
FEX	FEX	Piecewise	Piecewise function FEX
LG	LG	Lag	Voltage transducer
LL	LL	LeadLag	Regulator
LA	LA	LagAntiWindup	Lag AW on VR
INT	INT	Integrator	Integrator
WF	WF	Washout	Stablizing circuit feedback
LVG	LVG	LVGate	
VR	VR	GainLimiter	

Setpoints

Name	Target	Info
vref	vref0	Initial reference voltage input

Config Fields in [EXAC2]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

EXAC4

IEEE Type AC4 excitation system model.

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
syn		Synchronous generator idx			mandatory
TR	T_R	Sensing time constant	0.010	*p.u.*	
VIMAX	V_{IMAX}	Max. input voltage	5		
VIMIN	V_{IMIN}	Min. input voltage	-0.100		
TC	T_C	Lead time constant in lead-lag	1	*p.u.*	
TB	T_B	Lag time constant in lead-lag	1	*p.u.*	
KA	K_A	Regulator gain	80		
TA	T_A	Lag time constant in regulator	0.040	*p.u.*	
VRMAX	V_{RMAX}	Maximum excitation limit	8	*p.u.*	
VRMIN	V_{RMIN}	Minimum excitation limit	0	*p.u.*	
KC	K_C	Reactive power compensation gain	0		
ue	u_e	effective online status	0		
Sn	S_m	Rated power from generator	0	*MVA*	
Vn	V_m	Rated voltage from generator	0	*kV*	
bus	bus	Bus idx of the generators	0		

Variables

Name	Symbol	Type	Description	Unit	Properties
LG_y	LG_y	State	State in lag transfer function		v_str
LL_x	LL_x	State	State in lead-lag		v_str
LR_y	LR_y	State	State in lag transfer function		v_str
omega	ω	ExtState	Generator speed		
v	v	Algeb	Input to exciter (bus v or Eterm)		v_str
vout	$vout$	Algeb	Exciter final output voltage		v_str
vi	vi	Algeb	Total input voltages	*p.u.*	v_str
LL_y	LL_y	Algeb	Output of lead-lag		v_str
vfmax	$vfmax$	Algeb	Upper bound of output limiter		v_str
vfmin	$vfmin$	Algeb	Lower bound of output limiter		v_str
vf	vf	ExtAlgeb	Excitation field voltage to generator		
XadIfd	$XadIfd$	ExtAlgeb	Armature excitation current		
a	a	ExtAlgeb	Bus voltage phase angle		
vbus	$vbus$	ExtAlgeb	Bus voltage magnitude		

Initialization Equations

Name	Symbol	Type	Initial Value
LG_y	LG_y	State	v
LL_x	LL_x	State	$HLI_{zi}vi + HLI_{zl}VIMIN + HLI_{zu}VIMAX$
LR_y	LR_y	State	$KALL_y$
omega	ω	ExtState	
v	v	Algeb	$vbus$
vout	$vout$	Algeb	$uevf_0$
vi	vi	Algeb	$\frac{vf_0}{KA}$
LL_y	LL_y	Algeb	$HLI_{zi}vi + HLI_{zl}VIMIN + HLI_{zu}VIMAX$
vfmax	$vfmax$	Algeb	$-KCXadIfd + VRMAX$
vfmin	$vfmin$	Algeb	$-KCXadIfd + VRMIN$
vf	vf	ExtAlgeb	
XadIfd	$XadIfd$	ExtAlgeb	
a	a	ExtAlgeb	
vbus	$vbus$	ExtAlgeb	

Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
LG_y	LG_y	State	$-LG_y + v$	T_R
LL_x	LL_x	State	$HLI_{zi}vi + HLI_{zl}VIMIN + HLI_{zu}VIMAX - LL_x$	T_B
LR_y	LR_y	State	$KALL_y - LR_y$	T_A
omega	ω	ExtState	0	

Algebraic Equations

Nam	Symbol	Type	RHS of Equation "0 = g(x, y)"
v	v	Algeb	$-v + vbus$
vout	$vout$	Algeb	$ue\left(HLR_{zi}LR_y + HLR_{zl}vfmin + HLR_{zu}vfmax\right) - vout$
vi	vi	Algeb	$-LG_y - vi + vref_0$
LL_y	LL_y	Algeb	$LL_{LTz1}\left(-HLI_{zi}vi - HLI_{zl}VIMIN - HLI_{zu}VIMAX + LL_y\right) + LL_xTB - LL_yTB + TC\left(HLI_{zi}vi + HLI_{zl}VIMIN + HLI_{zu}VIMAX - LL_x\right)$
vfmax	$vfma$	Algeb	$-KCXadIfd + VRMAX - vfmax$
vfmin	$vfmi$	Algeb	$-KCXadIfd + VRMIN - vfmin$
vf	vf	ExtAlgeb	$ue\left(-vf_0 + vout\right)$
XadIfd	$XadI$	ExtAlgeb	0
a	a	ExtAlgeb	0
vbus	$vbus$	ExtAlgeb	0

Services

Name	Symbol	Equation	Type
vref0	V_{ref0}	$v + \frac{vf_0}{KA}$	PostInitService

Discretes

Name	Symbol	Type	Info
HLI	HLI	HardLimiter	Hard limiter on input
LL_LT	LT_{LL}	LessThan	
HLR	HLR	HardLimiter	Hard limiter on regulator output

Blocks

Name	Symbol	Type	Info
LG	LG	Lag	Sensing delay
LL	LL	LeadLag	Lead-lag compensator
LR	LR	Lag	Regulator

Setpoints

Name	Target	Info
vref	vref0	Const reference voltage

Config Fields in [EXAC4]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

ESST4B

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
syn		Synchronous generator idx			mandatory
TR	T_R	Sensing time constant	0.010	*p.u.*	
KPR	K_{PR}	Proportional gain 1	1	*p.u.*	
KIR	K_{IR}	Integral gain 1	0	*p.u.*	
VRMAX	V_{RMAX}	Maximum regulator limit	8	*p.u.*	
VRMIN	V_{RMIN}	Minimum regulator limit	0	*p.u.*	
TA	T_A	Lag time constant	0.100		
KPM	K_{PM}	Proportional gain 2	1	*p.u.*	
KIM	K_{IM}	Integral gain 2	0	*p.u.*	
VMMAX	V_{RMAX}	Maximum inner loop limit	8	*p.u.*	
VMMIN	V_{RMIN}	Minimum inner loop limit	0	*p.u.*	
KG	K_G	Feedback gain of inner field regulator	1		
KP	K_P	Potential circuit gain coeff.	4		
KI	K_I	Potential circuit gain coeff.	0.100		
VBMAX	V_{BMAX}	VB upper limit	18	*p.u.*	
KC	K_C	Rectifier loading factor proportional to commutating reactance	0.100		
XL	X_L	Potential source reactance	0.010		
THETAP	θ_P	Rectifier firing angle	0	*degree*	
VGMAX	V_{GMAX}	VG upper limit	20	*p.u.*	
ue	u_e	effective online status	0		
Sn	S_m	Rated power from generator	0	*MVA*	
Vn	V_m	Rated voltage from generator	0	*kV*	
bus	bus	Bus idx of the generators	0		

Variables

Name	Symbol	Type	Description	Unit	Properties
LG_y	LG_y	State	State in lag transfer function		v_str
PI1_xi	$\pi_{1\xi}$	State	Integrator output		v_str
LA_y	LA_y	State	State in lag transfer function		v_str
PI2_xi	$\pi_{2\xi}$	State	Integrator output		v_str
omega	ω	ExtState	Generator speed		
v	v	Algeb	Input to exciter (bus v or Eterm)		v_str
vout	$vout$	Algeb	Exciter final output voltage		v_str
UEL	UEL	Algeb	Interface var for under exc. limiter		v_str
IN	IN	Algeb	Input to FEX		v_str
FEX_y	FEX_y	Algeb	Output of piecewise		v_str
VB_x	VB_x	Algeb	Value before limiter		v_str
VG_x	VG_x	Algeb	Value before limiter		v_str
vref	$vref$	Algeb	Reference voltage input	*p.u.*	v_str
vi	vi	Algeb	Total input voltages	*p.u.*	v_str
PI1_ys	π_{1ys}	Algeb	PI summation before limit		v_str
PI1_y	π_{1y}	Algeb	PI output		v_str
PI2_ys	π_{2ys}	Algeb	PI summation before limit		v_str
PI2_y	π_{2y}	Algeb	PI output		v_str
vf	vf	ExtAlgeb	Excitation field voltage to generator		
XadIfd	$XadIfd$	ExtAlgeb	Armature excitation current		
a	a	ExtAlgeb	Bus voltage phase angle		
vbus	$vbus$	ExtAlgeb	Bus voltage magnitude		
vd	vd	ExtAlgeb	d-axis machine voltage		
vq	vq	ExtAlgeb	q-axis machine voltage		
Id	Id	ExtAlgeb	d-axis machine current		
Iq	Iq	ExtAlgeb	q-axis machine current		

Initialization Equations

Name	Symbol	Type	Initial Value
LG_y	LG_y	State	v
PI1_x	$\pi_{1\xi}$	State	$VGMAXVG_{limzu} + VG_{limzi}VG_x$
LA_y	LA_y	State	$1.0\pi_{1y}$
PI2_x	$\pi_{2\xi}$	State	$\text{safe}_{div}\left(vf_0, VBMAXVB_{limzu} + VB_{limzi}VB_x\right)$
omega	ω	ExtStat	
v	v	Algeb	$vbus$
vout	$vout$	Algeb	$uevf_0$
UEL	UEL	Algeb	0
IN	IN	Algeb	$\text{safe}_{div}\left(KCXadIfd, VE\right)$
FEX_	FEX_y	Algeb	$\text{FixPiecewise}\left(\left(1,\ IN \leq 0\right), \left(1 - 0.577IN,\ IN \leq 0.433\right), \left(\sqrt{0.75 - IN^2},\ IN \leq 0.75\right), (1.732$
VB_x	VB_x	Algeb	FEX_yVE
VG_x	VG_x	Algeb	$KGvout$
vref	$vref$	Algeb	v
vi	vi	Algeb	$-v + vref$
PI1_y	π_{1ys}	Algeb	$KPRvi + VGMAXVG_{limzu} + VG_{limzi}VG_x$
PI1_y	π_{1y}	Algeb	$\pi_{1limzi}\pi_{1ys} + \pi_{1limzl}VRMIN + \pi_{1limzu}VRMAX$
PI2_y	π_{2ys}	Algeb	$KPM\left(LA_y - VGMAXVG_{limzu} - VG_{limzi}VG_x\right) + \text{safe}_{div}\left(vf_0, VBMAXVB_{limzu} + VB_{limzi}VB_x\right)$
PI2_y	π_{2y}	Algeb	$\pi_{2limzi}\pi_{2ys} + \pi_{2limzl}VMMIN + \pi_{2limzu}VMMAX$
vf	vf	ExtAlgeb	
XadIfd	$XadIfd$	ExtAlgeb	
a	a	ExtAlgeb	
vbus	$vbus$	ExtAlgeb	
vd	vd	ExtAlgeb	
vq	vq	Ex-	

Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
LG_y	LG_y	State	$-LG_y + v$	T_R
PI1_xi	$\pi_{1\xi}$	State	$KIR\left(2\pi_{1y} - 2\pi_{1ys} + vi\right)$	
LA_y	LA_y	State	$-LA_y + 1.0\pi_{1y}$	T_A
PI2_xi	$\pi_{2\xi}$	State	$KIM\left(LA_y + 2\pi_{2y} - 2\pi_{2ys} - VGMAXVG_{limzu} - VG_{limzi}V\right.$	
omega	ω	ExtState	0	

Algebraic Equations

Nam	Sym-bol	Type	RHS of Equation "0 = g(x, y)"
v	v	Al-geb	$-v + vbus$
vout	$vout$	Al-geb	$\pi_{2y} ue \left(VBMAXVB_{limzu} + VB_{limzi} VB_x\right) - vout$
UEL	UEL	Al-geb	$-UEL$
IN	IN	Al-geb	$ue \left(-INVE + KCXadIfd\right)$
FEX_	$FEX_{}$	Al-geb	$-FEX_y$+FixPiecewise $\Big(\left(1,\ IN \leq 0\right), \left(1 - 0.577IN,\ IN \leq 0.433\right), \Big(\sqrt{0.75 - IN^2},\ IN \leq 0.75$
VB_x	VB_x	Al-geb	$FEX_y VE - VB_x$
VG_x	VG_x	Al-geb	$KGvout - VG_x$
vref	$vref$	Al-geb	$-vref + vref_0$
vi	vi	Al-geb	$-LG_y - vi + vref$
PI1_y	π_{1ys}	Al-geb	$KPRvi + \pi_{1\xi} - \pi_{1ys}$
PI1_y	π_{1y}	Al-geb	$\pi_{1limzi} \pi_{1ys} + \pi_{1limzl} VRMIN + \pi_{1limzu} VRMAX - \pi_{1y}$
PI2_y	π_{2ys}	Al-geb	$KPM \left(LA_y - VGMAXVG_{limzu} - VG_{limzi} VG_x\right) + \pi_{2\xi} - \pi_{2ys}$
PI2_y	π_{2y}	Al-geb	$\pi_{2limzi} \pi_{2ys} + \pi_{2limzl} VMMIN + \pi_{2limzu} VMMAX - \pi_{2y}$
vf	vf	Ex-tAl-geb	$ue \left(-vf_0 + vout\right)$
Xad-Ifd	$XadI$	Ex-tAl-geb	0
a	a	Ex-tAl-geb	0
vbus	$vbus$	Ex-tAl-geb	0
vd	vd	Ex-tAl-geb	0
vq	vq	Ex-tAl-geb	0
Id	Id	Ex-tAl-geb	0

Services

Name	Symbol	Equation	Type
KPC	K_{PC}	$KPe^{i \text{ radians } (THETAP)}$	ConstService
VE	V_E	$\lvert KPC\,(vd + ivq) + i\,(Id + iIq)\,(KI + KPCXL)\rvert$	VarService
vref0	V_{ref0}	$vref$	PostInitService

Discretes

Name	Symbol	Type	Info
VB_lim	lim_{VB}	HardLimiter	
VG_lim	lim_{VG}	HardLimiter	
PI1_lim	lim_{PI1}	HardLimiter	
PI2_lim	lim_{PI2}	HardLimiter	

Blocks

Name	Symbol	Type	Info
LG	LG	Lag	Voltage transducer
FEX	FEX	Piecewise	Piecewise function FEX
VB	VB	GainLimiter	VB with limiter
VG	VG	GainLimiter	Feedback gain with HL
PI1	$PI1$	PITrackAW	
LA	LA	Lag	Regulation delay
PI2	$PI2$	PITrackAW	

Setpoints

Name	Target	Info
vref	vref0	Const reference voltage

Config Fields in [ESST4B]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)
ksr	K_{sr}	2	Tracking gain for outer PI controller	
ksm	K_{sm}	2	Tracking gain for inner PI controller	

AC8B

Exciter AC8B model.

Reference:[1],[2]

[1] Powerworld, Exciter AC8B, [Online], Available: https://www.powerworld.com/WebHelp/Content/TransientModels_HTML/Exciter%20AC8B.htm

[2] NEPLAN, Exciters Models, [Online], Available: https://www.neplan.ch/wp-content/uploads/2015/08/Nep_EXCITERS1.pdf

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
syn		Synchronous generator idx			mandatory
TR	T_R	Sensing time constant	0.010	*p.u.*	
kP	k_P	PID proportional coeff.	10		
kI	k_I	PID integrative coeff.	10		
kD	k_D	PID derivative coeff.	10		
Td	T_d	PID derivative time constant.	0.200		
VPMAX	V_{PMAX}	PID maximum limit	999	*p.u.*	
VPMIN	V_{PMIN}	PID minimum limit	-999	*p.u.*	
VRMAX	V_{RMAX}	Maximum regulator limit	7.300	*p.u.*	
VRMIN	V_{RMIN}	Minimum regulator limit	1	*p.u.*	
VFEMAX	V_{FEMAX}	Exciter field current limit	999	*p.u.*	
VEMIN	V_{EMIN}	Minimum exciter voltage output	-999	*p.u.*	
TA	T_A	Lag time constant in anti-windup lag	0.040	*p.u.*	
KA	K_A	Gain in anti-windup lag TF	40	*p.u.*	
TE	T_E	Exciter integrator time constant	0.800	*p.u.*	
E1	E_1	First saturation point	0	*p.u.*	
SE1	S_{E1}	Value at first saturation point	0	*p.u.*	
E2	E_2	Second saturation point	1	*p.u.*	
SE2	S_{E2}	Value at second saturation point	1	*p.u.*	
KE	K_E	Gain added to saturation	1	*p.u.*	
KD	K_D	Ifd feedback gain	0		
KC	K_C	Rectifier loading factor proportional to commutating reactance	0.100		
ue	u_e	effective online status	0		
Sn	S_m	Rated power from generator	0	*MVA*	
Vn	V_m	Rated voltage from generator	0	*kV*	
bus	bus	Bus idx of the generators	0		

Variables

Name	Symbol	Type	Description	Unit	Properties
LG_y	LG_y	State	State in lag transfer function		v_str
PID_xi	PID_ξ	State	Integrator output		v_str
PID_WO_x	PID_{WOx}	State	State in washout filter		v_str
LA_y	LA_y	State	State in lag TF		v_str
INT_y	INT_y	State	Integrator output		v_str,v_iter
omega	ω	ExtState	Generator speed		
v	v	Algeb	Input to exciter (bus v or Eterm)		v_str
vout	$vout$	Algeb	Exciter final output voltage		v_str
IN	IN	Algeb	Input to FEX		v_str,v_iter
FEX_y	FEX_y	Algeb	Output of piecewise		v_str,v_iter
UEL	UEL	Algeb	Interface var for under exc. limiter		v_str
OEL	OEL	Algeb	Interface var for over exc. limiter		v_str
Vs	Vs	Algeb	Voltage compensation from PSS		v_str
vref	$vref$	Algeb	Reference voltage input	*p.u.*	v_str
vi	vi	Algeb	Total input voltages	*p.u.*	v_str
PID_uin	PID_{uin}	Algeb	PID input		v_str
PID_WO_y	PID_{WOy}	Algeb	Output of washout filter		v_str
PID_ys	PID_{ys}	Algeb	PI summation before limit		v_str
PID_y	PID_y	Algeb	PI output		v_str
Se	Se	Algeb	saturation output		v_str
VFE	VFE	Algeb	Combined saturation feedback	*p.u.*	v_str
vf	vf	ExtAlgeb	Excitation field voltage to generator		
XadIfd	$XadIfd$	ExtAlgeb	Armature excitation current		
a	a	ExtAlgeb	Bus voltage phase angle		
vbus	$vbus$	ExtAlgeb	Bus voltage magnitude		

Initialization Equations

Name	Symbol	Type	Initial Value
LG_y	LG_y	State	v
PID_xi	PID_ξ	State	$\frac{VFE}{KA}$
PID_W	PID_W	State	PID_{uin}
LA_y	LA_y	State	$KAPID_y$
INT_y	INT_y	State	$FEX_yINT_y - vf_0$
omega	ω	ExtSta	
v	v	Algeb	$vbus$
vout	$vout$	Algeb	$uevf_0$
IN	IN	Algeb	$-ININT_y + KCXadIfd$
FEX_y	FEX_y	Algeb	$-FEX_y$+FixPiecewise $\left((1,\ IN \leq 0), (1 - 0.577IN,\ IN \leq 0.433), \left(\sqrt{0.75 - IN^2},\ IN \leq 0\right.\right.$
UEL	UEL	Algeb	UEL_0
OEL	OEL	Algeb	OEL_0
Vs	Vs	Algeb	0
vref	$vref$	Algeb	v
vi	vi	Algeb	$-v + vref$
PID_uin	PID_{uin}	Algeb	vi
PID_W	PID_W	Algeb	0
PID_ys	PID_{ys}	Algeb	$kPvi + \frac{VFE}{KA}$
PID_y	PID_y	Algeb	$PID_{limzi}PID_{ys} + PID_{limzl}VPMIN + PID_{limzu}VPMAX$
Se	Se	Algeb	$SAT_B\left(INT_y - SAT_A\right)^2$ Indicator $\left(INT_y > SAT_A\right)$
VFE	VFE	Algeb	$INT_yKE + KDXadIfd + Se$
vf	vf	ExtAlgeb	
XadIfd	$XadIfd$	ExtAlgeb	
a	a	ExtAlgeb	
vbus	$vbus$	ExtAl-	

Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
LG_y	LG_y	State	$-LG_y + v$	T_R
PID_xi	PID_ξ	State	$kI\left(2PID_y - 2PID_{ys} + vi\right)$	
PID_WO_x	PID_{WOx}	State	$-PID_{WOx} + PID_{uin}$	T_d
LA_y	LA_y	State	$KAPID_y - LA_y$	T_A
INT_y	INT_y	State	$ue\left(LA_y - VFE\right)$	T_E
omega	ω	ExtState	0	

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
v	v	Algeb	$-v + vbus$
vout	$vout$	Algeb	$ue\left(FEX_y INT_y - vout\right)$
IN	IN	Algeb	$ue\left(-ININT_y + KCXadIfd\right)$
FEX_y	FEX_y	Algeb	$-FEX_y$+FixPiecewise $\Big((1,\ IN \leq 0), (1 - 0.577IN,\ IN \leq 0.433), \Big(\sqrt{0.75 - IN^2},\ IN \leq 0$
UEL	UEL	Algeb	$-UEL + UEL_0$
OEL	OEL	Algeb	$-OEL + OEL_0$
Vs	Vs	Algeb	$-Vs$
vref	$vref$	Algeb	$-vref + vref_0$
vi	vi	Algeb	$ue\left(-LG_y + OEL + UEL + Vs - vi + vref\right)$
PID_uin	PID_{uin}	Algeb	$-PID_{uin} + vi$
PID_WO	PID_{WO}	Algeb	$-PID_{WOy}Td + kD\left(-PID_{WOx} + PID_{uin}\right)$
PID_ys	PID_{ys}	Algeb	$PID_{WOy} + PID_{\xi} - PID_{ys} + kPvi$
PID_y	PID_y	Algeb	$PID_{limzi}PID_{ys} + PID_{limzl}VPMIN + PID_{limzu}VPMAX - PID_y$
Se	Se	Algeb	$ue\left(SAT_B SL_{z0}\left(INT_y - SAT_A\right)^2 - Se\right)$
VFE	VFE	Algeb	$ue\left(INT_y KE + KDXadIfd + Se - VFE\right)$
vf	vf	ExtAlgeb	$ue\left(-vf_0 + vout\right)$
XadIfd	$XadIfd$	ExtAlgeb	0
a	a	ExtAlgeb	0
vbus	$vbus$	ExtAlgeb	0

Services

Name	Symbol	Equation	Type
UEL0	$UEL0$	0	ConstService
OEL0	$OEL0$	0	ConstService
SAT_E1	E_{SAT}^{1c}	E_1	ConstService
SAT_E2	E_{SAT}^{2c}	$E_2 - 2SAT_{zSE2} + 2$	ConstService
SAT_SE1	SE_{SAT}^{1c}	SE_1	ConstService
SAT_SE2	SE_{SAT}^{2c}	$-2SAT_{zSE2} + SE_2 + 2$	ConstService
SAT_a	a_{SAT}	$\sqrt{\frac{SAT_{E1}SAT_{SE1}}{SAT_{E2}SAT_{SE2}}}$ (Indicator $(SAT_{SE2} > 0)$ + Indicator $(SAT_{SE2} <$	ConstService
SAT_A	A_{SAT}^{q}	$SAT_{E2} - \frac{SAT_{E1} - SAT_{E2}}{SAT_a - 1}$	ConstService
SAT_B	B_{SAT}^{q}	$\frac{SAT_{E2}SAT_{SE2}(SAT_a - 1)^2(\text{Indicator}\,(SAT_a > 0) + \text{Indicator}\,(SAT_a < 0))}{(SAT_{E1} - SAT_{E2})^2}$	ConstService
vref0	V_{ref0}	v	PostInitService

Discretes

Name	Symbol	Type	Info
PID_lim	lim_{PID}	HardLimiter	
LA_lim	lim_{LA}	AntiWindup	Limiter in Lag
SL	SL	LessThan	

Blocks

Name	Symbol	Type	Info
FEX	FEX	Piecewise	Piecewise function FEX
LG	LG	Lag	Voltage transducer
PID	PID	PIDTrackAW	PID
PID_WO	$PID\,WO_{PID}$	Washout	Washout
LA	LA	LagAntiWindup	V_{R}, Anti-windup lag
SAT	SAT	ExcQuadSat	Field voltage saturation
INT	INT	Integrator	V_E, integrator

Setpoints

Name	Target	Info
vref	vref0	Initial reference voltage input

Config Fields in [AC8B]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)
ks		2	Tracking gain for PID controller	

IEEET3

Exciter IEEET3.

Reference:

[1] PowerWorld, Exciter IEEET3, [Online],

[2] NEPLAN, Exciters Models, [Online],

Available:

https://www.powerworld.com/WebHelp/Content/TransientModels_HTML/Exciter%20IEEET3.htm

https://www.neplan.ch/wp-content/uploads/2015/08/Nep_EXCITERS1.pdf

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
syn		Synchronous generator idx			mandatory
TR	T_R	Sensing time constant	0.020	*p.u.*	
KA	K_A	Regulator gain	5	*p.u.*	
TA	T_A	Lag time constant in anti-windup lag	0.040	*p.u.*	
VRMAX	V_{RMAX}	Maximum regulator limit	7.300	*p.u.*	
VRMIN	V_{RMIN}	Minimum regulator limit	-7.300	*p.u.*	
VBMAX	V_{BMAX}	VB upper limit	18	*p.u.*	
KE	K_E	Exciter integrator constant	1	*p.u.*	
TE	T_E	Exciter integrator time constant	1	*p.u.*	
KF	K_F	Feedback gain	0.100		
TF	T_F	Feedback delay	1		non_zero,non_negative
KP	K_P	Potential circuit gain coeff.	4		
KI	K_I	Potential circuit gain coeff.	0.100		
ue	u_e	effective online status	0		
Sn	S_m	Rated power from generator	0	*MVA*	
Vn	V_m	Rated voltage from generator	0	*kV*	
bus	bus	Bus idx of the generators	0		

Variables

Name	Symbol	Type	Description	Unit	Properties
LG_y	LG_y	State	State in lag transfer function		v_str
LA3_y	LA_{3y}	State	State in lag TF		v_str
LA1_y	LA_{1y}	State	State in lag transfer function		v_str
WF_x	WF_x	State	State in washout filter		v_str
omega	ω	ExtState	Generator speed		
v	v	Algeb	Input to exciter (bus v or Eterm)		v_str
vout	$vout$	Algeb	Exciter final output voltage		v_str
UEL	UEL	Algeb	Interface var for under exc. limiter		v_str
OEL	OEL	Algeb	Interface var for over exc. limiter		v_str
Vs	Vs	Algeb	Voltage compensation from PSS		v_str
vref	$vref$	Algeb	Reference voltage input	*p.u.*	v_str
vi	vi	Algeb	Total input voltages	*p.u.*	v_str
WF_y	WF_y	Algeb	Output of washout filter		v_str
SQE	SQE	Algeb	Square of error after mul		v_str
VB_y	VB_y	Algeb	Output of piecewise		v_str
vf	vf	ExtAlgeb	Excitation field voltage to generator		
XadIfd	$XadIfd$	ExtAlgeb	Armature excitation current		
a	a	ExtAlgeb	Bus voltage phase angle		
vbus	$vbus$	ExtAlgeb	Bus voltage magnitude		
vd	vd	ExtAlgeb	d-axis machine voltage		
vq	vq	ExtAlgeb	q-axis machine voltage		
Id	Id	ExtAlgeb	d-axis machine current		
Iq	Iq	ExtAlgeb	q-axis machine current		

Initialization Equations

Name	Symbol	Type	Initial Value
LG_y	LG_y	State	v
LA3_y	LA_{3y}	State	$KAue\left(-WF_y + vi\right)$
LA1_y	LA_{1y}	State	$\frac{ue(HL_{zi}VB_y + HL_{zu}VBMAX)}{KE}$
WF_x	WF_x	State	LA_{1y}
omega	ω	ExtState	
v	v	Algeb	$vbus$
vout	$vout$	Algeb	$uevf_0$
UEL	UEL	Algeb	UEL_0
OEL	OEL	Algeb	OEL_0
Vs	Vs	Algeb	0
vref	$vref$	Algeb	$v + vb_0$
vi	vi	Algeb	$-v + vref$
WF_y	WF_y	Algeb	0
SQE	SQE	Algeb	$VE^2 - 0.6084XadIfd^2$
VB_y	VB_y	Algeb	FixPiecewise $\left((LA_{3y}ue,\ SQE \leq 0), \left(ue\left(LA_{3y} + \sqrt{SQE}\right),\ \text{True}\right)\right)$
vf	vf	ExtAlgeb	
XadIfd	$XadIfd$	ExtAlgeb	
a	a	ExtAlgeb	
vbus	$vbus$	ExtAlgeb	
vd	vd	ExtAlgeb	
vq	vq	ExtAlgeb	
Id	Id	ExtAlgeb	
Iq	Iq	ExtAlgeb	

Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
LG_y	LG_y	State	$-LG_y + v$	T_R
LA3_y	LA_{3y}	State	$KAue\left(-WF_y + vi\right) - LA_{3y}$	T_A
LA1_y	LA_{1y}	State	$-KELA_{1y} + ue\left(HL_{zi}VB_y + HL_{zu}VBMAX\right)$	T_E
WF_x	WF_x	State	$LA_{1y} - WF_x$	T_F
omega	ω	ExtState	0	

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
v	v	Algeb	$-v + vbus$
vout	$vout$	Algeb	$ue\left(LA_{1y} - vout\right)$
UEL	UEL	Algeb	$-UEL + UEL_0$
OEL	OEL	Algeb	$-OEL + OEL_0$
Vs	Vs	Algeb	$-Vs$
vref	$vref$	Algeb	$-vref + vref_0$
vi	vi	Algeb	$ue\left(-LG_y + OEL + UEL + Vs - vi + vref\right)$
WF_y	WF_y	Algeb	$KF\left(LA_{1y} - WF_x\right) - TFWF_y$
SQE	SQE	Algeb	$-SQE + VE^2 - 0.6084XadIfd^2$
VB_y	VB_y	Algeb	$-VB_y$+FixPiecewise $\left(\left(LA_{3y}ue,\ SQE \leq 0\right), \left(ue\left(LA_{3y} + \sqrt{SQE}\right),\ \text{True}\right)\right)$
vf	vf	ExtAlgeb	$ue\left(-vf_0 + vout\right)$
XadIfd	$XadIfd$	ExtAlgeb	0
a	a	ExtAlgeb	0
vbus	$vbus$	ExtAlgeb	0
vd	vd	ExtAlgeb	0
vq	vq	ExtAlgeb	0
Id	Id	ExtAlgeb	0
Iq	Iq	ExtAlgeb	0

Services

Name	Symbol	Equation	Type
VE	V_E	$\sqrt{Id^2KI^2 + 2IdKIKPvq + Iq^2KI^2 - 2IqKIKPvd + KP^2va}$	VarService
V40	$V40$	$\sqrt{VE^2 - 0.6084XadIfd^2}$	ConstService
VR0	V_{R0}	$KEvf_0 - V_{40}$	ConstService
vb0	V_{b0}	$\frac{VR_0}{KA}$	ConstService
VRMAXc	$VRMAXc$	$VRMAX - 999_zVRM + 999$	ConstService
UEL0	$UEL0$	0	ConstService
OEL0	$OEL0$	0	ConstService
vref0	V_{ref0}	$vref$	PostInitService
zeros	$zeros$	0.0	ConstService

Discretes

Name	Symbol	Type	Info
LA3_lim	lim_{LA3}	AntiWindup	Limiter in Lag
SL	SL	LessThan	
HL	HL	HardLimiter	Hard limiter for VB

Blocks

Name	Symbol	Type	Info
LG	LG	Lag	Sensing delay
LA3	$LA3$	LagAntiWindup	V_{R}, Lag Anti-Windup
LA1	$LA1$	Lag	
WF	WF	Washout	V_F, stablizing circuit feedback, washout
VB	VB	Piecewise	

Setpoints

Name	Target	Info
vref	vref0	Constant vref

Config Fields in [IEEET3]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

ESAC1A

Exciter ESAC1A.

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
syn		Synchronous generator idx			mandatory
TR	T_R	Sensing time constant	0.010	*p.u.*	
TB	T_B	Lag time constant in lead-lag	1	*p.u.*	non_negative
TC	T_C	Lead time constant in lead-lag	1	*p.u.*	non_negative
VAMAX	V_{AMAX}	V_A upper limit	999	*p.u.*	
VAMIN	V_{AMIN}	V_A lower limit	-999	*p.u.*	
KA	K_A	Regulator gain	80		
TA	T_A	Lag time constant in regulator	0.040	*p.u.*	non_negative
VRMAX	V_{RMAX}	Max. exc. limit (0-unlimited)	7.300	*p.u.*	
VRMIN	V_{RMIN}	Min. excitation limit	-7.300	*p.u.*	
TE	T_E	Integrator time constant	0.800	*p.u.*	non_negative
E1	E_1	First saturation point	0	*p.u.*	
SE1	S_{E1}	Value at first saturation point	0	*p.u.*	
E2	E_2	Second saturation point	1	*p.u.*	
SE2	S_{E2}	Value at second saturation point	1	*p.u.*	
KC	K_C	Rectifier loading factor proportional to commutating reactance	0.100		
KD	K_D	Ifd feedback gain	0		
KE	K_E	Gain added to saturation	1		
KF	K_F	Feedback gain	0.100		
TF	T_{F1}	Feedback washout time constant	1	*p.u.*	non_zero,non_negative
Switch	S_w	Switch that PSS/E did not implement	0	*bool*	
ue	u_e	effective online status	0		
Sn	S_m	Rated power from generator	0	*MVA*	
Vn	V_m	Rated voltage from generator	0	*kV*	
bus	bus	Bus idx of the generators	0		

Variables

Name	Symbol	Type	Description	Unit	Properties
LG_y	LG_y	State	State in lag transfer function		v_str
LL_x	LL_x	State	State in lead-lag		v_str
LA_y	LA_y	State	State in lag TF		v_str
INT_y	INT_y	State	Integrator output		v_str,v_iter
WF_x	WF_x	State	State in washout filter		v_str
omega	ω	ExtState	Generator speed		
v	v	Algeb	Input to exciter (bus v or Eterm)		v_str
vout	$vout$	Algeb	Exciter final output voltage		v_str
UEL	UEL	Algeb	Interface var for under exc. limiter		v_str
OEL	OEL	Algeb	Interface var for over exc. limiter		v_str
Vs	Vs	Algeb	Voltage compensation from PSS		v_str
vref	$vref$	Algeb	Reference voltage input	*p.u.*	v_str
IN	IN	Algeb	Input to FEX		v_str,v_iter
FEX_y	FEX_y	Algeb	Output of piecewise		v_str,v_iter
vi	vi	Algeb	Total input voltages	*p.u.*	v_str
LL_y	LL_y	Algeb	Output of lead-lag		v_str
Se	Se	Algeb	saturation output		v_str
VFE	VFE	Algeb	Combined saturation feedback	*p.u.*	v_str
WF_y	WF_y	Algeb	Output of washout filter		v_str
vf	vf	ExtAlgeb	Excitation field voltage to generator		
XadIfd	$XadIfd$	ExtAlgeb	Armature excitation current		
a	a	ExtAlgeb	Bus voltage phase angle		
vbus	$vbus$	ExtAlgeb	Bus voltage magnitude		

Initialization Equations

Nam	Sym-bol	Type	Initial Value
LG_y	LG_y	State	v
LL_x	LL_x	State	vi
LA_y	LA_y	State	$KALL_y$
INT_	INT_y	State	$FEX_yINT_y - vf_0$
WF_	WF_x	State	VFE
omeg	ω	ExtSta	
v	v	Al-geb	$vbus$
vout	$vout$	Al-geb	$uevf_0$
UEL	UEL	Al-geb	UEL_0
OEL	OEL	Al-geb	OEL_0
Vs	Vs	Al-geb	0
vref	$vref$	Al-geb	$v + \frac{VFE}{KA}$
IN	IN	Al-geb	$-ININT_y + KCXadIfd$
FEX_	$FEX_$	Al-geb	$-FEX_y$+FixPiecewise $\Big((1,\ IN \leq 0), (1 - 0.577IN,\ IN \leq 0.433), \Big(\sqrt{0.75 - IN^2},\ IN \leq 0.7$
vi	vi	Al-geb	$-v + vref$
LL_y	LL_y	Al-geb	vi
Se	Se	Al-geb	$SAT_B\left(INT_y - SAT_A\right)^2$ Indicator $\left(INT_y > SAT_A\right)$
VFE	VFE	Al-geb	$INT_yKE + KDXadIfd + Se$
WF_	WF_y	Al-geb	0
vf	vf	Ex-tAl-geb	
Xad-Ifd	$XadI$	Ex-tAl-geb	
a	a	Ex-tAl-geb	
vbus	$vbus$	Ex-tAl-geb	

Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
LG_y	LG_y	State	$-LG_y + v$	T_R
LL_x	LL_x	State	$-LL_x + vi$	T_B
LA_y	LA_y	State	$KALL_y - LA_y$	T_A
INT_y	INT_y	State	$ue\left(LVG_{ltz0}OEL + LVG_{ltz1}\left(HVG_{ltz0}UEL + HVG_{ltz1}LA_y\right)\right.$ -	T_E
WF_x	WF_x	State	$VFE - WF_x$	T_{F1}
omega	ω	ExtState	0	

Algebraic Equations

Nam	Sym-bol	Type	RHS of Equation "0 = g(x, y)"
v	v	Al-geb	$-v + vbus$
vout	$vout$	Al-geb	$FEX_y INT_y ue - vout$
UEL	UEL	Al-geb	$-UEL + UEL_0$
OEL	OEL	Al-geb	$-OEL + OEL_0$
Vs	Vs	Al-geb	$-Vs$
vref	$vref$	Al-geb	$-vref + vref_0$
IN	IN	Al-geb	$ue\left(-ININT_y + KCXadIfd\right)$
FEX_	$FEX_{}$	Al-geb	$-FEX_y$+FixPiecewise $\Big(\left(1,\ IN \leq 0\right), \left(1 - 0.577IN,\ IN \leq 0.433\right), \Big(\sqrt{0.75 - IN^2},\ IN \leq 0.7$
vi	vi	Al-geb	$ue\left(-LG_y + OEL + UEL + Vs - vi + vref\right)$
LL_y	LL_y	Al-geb	$LL_{LTz1}\left(LL_y - vi\right) + LL_x TB - LL_y TB + TC\left(-LL_x + vi\right)$
Se	Se	Al-geb	$ue\left(SAT_B SL_{z0}\left(INT_y - SAT_A\right)^2 - Se\right)$
VFE	VFE	Al-geb	$ue\left(INT_y KE + KDXadIfd + Se - VFE\right)$
WF_	WF_y	Al-geb	$KF\left(VFE - WF_x\right) - TFWF_y$
vf	vf	Ex-tAl-geb	$ue\left(-vf_0 + vout\right)$
Xad-Ifd	$XadI$	Ex-tAl-geb	0
a	a	Ex-tAl-geb	0
vbus	$vbus$	Ex-tAl-geb	0

Services

Name	Symbol	Equation	Type
UEL0	$UEL0$	-999	ConstService
OEL0	$OEL0$	999	ConstService
VAMAXu	$VAMAXu$	$VAMAXue - 999ue + 999$	ConstService
VAMINu	$VAMINu$	$VAMINue + 999ue - 999$	ConstService
SAT_E1	E_{SAT}^{1c}	E_1	ConstService
SAT_E2	E_{SAT}^{2c}	$E_2 - 2SAT_{zSE2} + 2$	ConstService
SAT_SE1	SE_{SAT}^{1c}	SE_1	ConstService
SAT_SE2	SE_{SAT}^{2c}	$-2SAT_{zSE2} + SE_2 + 2$	ConstService
SAT_a	a_{SAT}	$\sqrt{\frac{SAT_{E1}SAT_{SE1}}{SAT_{E2}SAT_{SE2}}}\left(\text{Indicator}\left(SAT_{SE2} > 0\right) + \text{Indicator}\left(SAT_{SE2}\right.\right.$	ConstService
SAT_A	A_{SAT}^{q}	$SAT_{E2} - \frac{SAT_{E1} - SAT_{E2}}{SAT_a - 1}$	ConstService
SAT_B	B_{SAT}^{q}	$\frac{SAT_{E2}SAT_{SE2}\left(SAT_a - 1\right)^2\left(\text{Indicator}\left(SAT_a > 0\right) + \text{Indicator}\left(SAT_a < 0\right)\right)}{\left(SAT_{E1} - SAT_{E2}\right)^2}$	ConstService
vref0	V_{ref0}	$vref$	PostInitService

Discretes

Name	Symbol	Type	Info
LL_LT	LT_{LL}	LessThan	
LA_lim	lim_{LA}	AntiWindup	Limiter in Lag
HVG_lt	$None_{HVG}$	LessThan	
LVG_lt	$None_{LVG}$	LessThan	
SL	SL	LessThan	

Blocks

Name	Symbol	Type	Info
FEX	FEX	Piecewise	Piecewise function FEX
LG	LG	Lag	Voltage transducer
LL	LL	LeadLag	V_A, Lead-lag compensator
LA	LA	LagAntiWindup	V_A, Anti-windup lag
HVG	HVG	HVGate	HVGate for under excitation
LVG	LVG	LVGate	HVGate for under excitation
SAT	SAT	ExcQuadSat	Field voltage saturation
INT	INT	Integrator	V_E, integrator
WF	WF	Washout	Stablizing circuit feedback

Setpoints

Name	Target	Info
vref	vref0	Initial reference voltage input

Config Fields in [ESAC1A]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

ESST1A

Exciter ESST1A model.

Reference:

[1] PowerWorld, Exciter ESST1A, [Online],

[2] NEPLAN, Exciters Models, [Online],

Available: https://www.powerworld.com/WebHelp/#TransientModels_HTML/Exciter%20ESST1A%20and%20ESST1A_GE.htm

https://www.neplan.ch/wp-content/uploads/2015/08/Nep_EXCITERS1.pdf

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
syn		Synchronous generator idx			mandatory
TR	T_R	Sensing time constant	0.010		
VIMAX	V_{IMAX}	Max. input voltage	0.800		
VIMIN	V_{IMIN}	Min. input voltage	-0.100		
TB	T_B	Lag time constant in lead-lag	1		
TC	T_C	Lead time constant in lead-lag	1		
TB1	T_{B1}	Lag time constant in lead-lag 1	1		
TC1	T_{C1}	Lead time constant in lead-lag 1	1		
VAMAX	V_{AMAX}	V_A upper limit	999	*p.u.*	
VAMIN	V_{AMIN}	V_A lower limit	-999	*p.u.*	
KA	K_A	Regulator gain	80		
TA	T_A	Lag time constant in regulator	0.040		
ILR	I_{LR}	Exciter output current limite reference	1		
KLR	K_{LR}	Exciter output current limiter gain	1		
VRMAX	V_{RMAX}	Maximum voltage regulator output limit	7.300	*p.u.*	
VRMIN	V_{RMIN}	Minimum voltage regulator output limit	-7.300	*p.u.*	
KF	K_F	Feedback gain	0.100		
TF	T_F	Feedback washout time constant	1		
KC	K_C	Rectifier loading factor proportional to commutating reactance	0.100		
UELc	UEL_c	Alternate UEL inputs, input code 1-3	1		
VOSc	VOS_c	Alternate Stabilizer inputs, input code 1-2	1		
ue	u_e	effective online status	0		
Sn	S_m	Rated power from generator	0	*MVA*	
Vn	V_m	Rated voltage from generator	0	*kV*	
bus	bus	Bus idx of the generators	0		

Variables

Name	Symbol	Type	Description	Unit	Properties
LG_y	LG_y	State	State in lag transfer function		v_str
LL_x	LL_x	State	State in lead-lag		v_str
LL1_x	LL_{1x}	State	State in lead-lag		v_str
LA_y	LA_y	State	State in lag TF		v_str
WF_x	WF_x	State	State in washout filter		v_str
omega	ω	ExtState	Generator speed		
v	v	Algeb	Input to exciter (bus v or Eterm)		v_str
vout	$vout$	Algeb	Exciter final output voltage		v_str
UEL	UEL	Algeb	Interface var for under exc. limiter		v_str
OEL	OEL	Algeb	Interface var for over exc. limiter		v_str
Vs	Vs	Algeb	Voltage compensation from PSS		v_str
vref	$vref$	Algeb	Reference voltage input	*p.u.*	v_str,v_iter
SG	SG	Algeb	SG		v_str
LR_x	LR_x	Algeb	Value before limiter		v_str
vi	vi	Algeb	Total input voltages	*p.u.*	v_str,v_iter
vil_x	vil_x	Algeb	Value before limiter		v_str
UEL2	UEL_2	Algeb	UEL_2 as HVG1 u1		v_str
LL_y	LL_y	Algeb	Output of lead-lag		v_str
LL1_y	LL_{1y}	Algeb	Output of lead-lag		v_str
vas	vas	Algeb	V_A after subtraction, as HVG u2		v_str,v_iter
UEL3	UEL_3	Algeb	UEL_3 as HVG u1		v_str
vol_x	vol_x	Algeb	Value before limiter		v_str
WF_y	WF_y	Algeb	Output of washout filter		v_str
vf	vf	ExtAlgeb	Excitation field voltage to generator		
XadIfd	$XadIfd$	ExtAlgeb	Armature excitation current		
a	a	ExtAlgeb	Bus voltage phase angle		
vbus	$vbus$	ExtAlgeb	Bus voltage magnitude		
vd	vd	ExtAlgeb	d-axis machine voltage		
vq	vq	ExtAlgeb	q-axis machine voltage		

Initialization Equations

Name	Symbol	Type	Initial Value
LG_y	LG_y	State	v
LL_x	LL_x	State	$HVG_{1ltz0}UEL_2+HVG_{1ltz1}\left(VIMAXvil_{limzu}+VIMINvil_{limzl}+vil_{limzi}vil_x\right)$
LL1_x	LL_{1x}	State	LL_y
LA_y	LA_y	State	$KALL_{1y}$
WF_x	WF_x	State	$LVG_{ltz0}OEL+LVG_{ltz1}\left(HVG_{ltz0}UEL_3+HVG_{ltz1}vas\right)$
omega	ω	ExtState	
v	v	Algeb	$vbus$
vout	$vout$	Algeb	$uevf_0$
UEL	UEL	Algeb	UEL_0
OEL	OEL	Algeb	OEL_0
Vs	Vs	Algeb	0
vref	$vref$	Algeb	$ue\left(-SGSWVOS_{s1}-SWUEL_{s1}UEL+v+\frac{LR_{limzi}LR_x+LR_{limzl}zero-SGSWVOS_{s2}+vf_0}{KA}\right)$
SG	SG	Algeb	SG_0
LR_x	LR_x	Algeb	$KLR\left(-ILR+XadIfd\right)$
vi	vi	Algeb	$ue\left(-LG_y+SGSWVOS_{s1}+SWUEL_{s1}UEL+Vs-WF_y+vref\right)$
vil_x	vil_x	Algeb	vi
UEL2	UEL_2	Algeb	$ue\left(SWUEL_{s2}UEL+llim\left(1-SWUEL_{s2}\right)\right)$
LL_y	LL_y	Algeb	$HVG_{1ltz0}UEL_2+HVG_{1ltz1}\left(VIMAXvil_{limzu}+VIMINvil_{limzl}+vil_{limzi}vil_x\right)$
LL1_y	LL_{1y}	Algeb	LL_y
vas	vas	Algeb	$ue\left(LA_y-LR_{limzi}LR_x-LR_{limzl}zero+SGSWVOS_{s2}\right)$
UEL3	UEL_3	Algeb	$ue\left(SWUEL_{s3}UEL+llim\left(1-SWUEL_{s3}\right)\right)$
vol_x	vol_x	Algeb	$LVG_{ltz0}OEL+LVG_{ltz1}\left(HVG_{ltz0}UEL_3+HVG_{ltz1}vas\right)$
WF_y	WF_y	Algeb	0
vf	vf	ExtAlgeb	
XadIfd	$XadIfd$	ExtAlgeb	
a	a	ExtAlgeb	
vbus	$vbus$	ExtAlgeb	
vd	vd	ExtAlgeb	
vq	vq	ExtAlgeb	

Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
LG_y	LG_y	State	$-LG_y + v$	T_R
LL_x	LL_x	State	$HVG_{1ltz0}UEL_2 + HVG_{1ltz1}\left(VIMAXvil_{limzu} + VIMINvil_{limzl} - vil_x\right) - LL_x$	T_B
LL1_x	LL_{1x}	State	$-LL_{1x} + LL_y$	T_{B1}
LA_y	LA_y	State	$KALL_{1y} - LA_y$	T_A
WF_x	WF_x	State	$LVG_{ltz0}OEL + LVG_{ltz1}\left(HVG_{ltz0}UEL_3 + HVG_{ltz1}vas\right) - WF_x$	T_F
omega	ω	ExtState	0	

Algebraic Equations

Nam	Sym bol	Type	RHS of Equation "0 = g(x, y)"
v	v	Al-geb	$-v + vbus$
vout	$vout$	Al-geb	$ue\left(efdlvol_{limzl} + efduvol_{limzu} + vol_{limzi}vol_x\right) - vout$
UEL	UEL	Al-geb	$-UEL + UEL_0$
OEL	OEL	Al-geb	$-OEL + OEL_0$
Vs	Vs	Al-geb	$-Vs$
vref	$vref$	Al-geb	$-vref + vref_0$
SG	SG	Al-geb	$-SG + SG_0$
LR_x	LR_x	Al-geb	$KLR\left(-ILR + XadIfd\right) - LR_x$
vi	vi	Al-geb	$ue\left(-LG_y + SGSWVOS_{s1} + SWUEL_{s1}UEL + Vs - WF_y + vref\right) - vi$
vil_x	vil_x	Al-geb	$vi - vil_x$
UEL	UEL	Al-geb	$-UEL_2 + ue\left(SWUEL_{s2}UEL + llim\left(1 - SWUEL_{s2}\right)\right)$
LL_y	LL_y	Al-geb	$LL_{LTz1}\left(-HVG_{1ltz0}UEL_2 - HVG_{1ltz1}\left(VIMAXvil_{limzu} + VIMINvil_{limzl} + vil_{limzi}vil_x\right) + L$ $LL_xTB - LL_yTB + TC\left(HVG_{1ltz0}UEL_2 + HVG_{1ltz1}\left(VIMAXvil_{limzu} + VIMINvil_{limzl} + vil\right.\right.$
LL1_	LL_{1y}	Al-geb	$LL_{1LTz1}\left(LL_{1y} - LL_y\right) + LL_{1x}TB_1 - LL_{1y}TB_1 + TC_1\left(-LL_{1x} + LL_y\right)$
vas	vas	Al-geb	$ue\left(LA_y - LR_{limzi}LR_x - LR_{limzl}zero + SGSWVOS_{s2}\right) - vas$
UEL	UEL	Al-geb	$-UEL_3 + ue\left(SWUEL_{s3}UEL + llim\left(1 - SWUEL_{s3}\right)\right)$
vol_x	vol_x	Al-geb	$LVG_{ltz0}OEL + LVG_{ltz1}\left(HVG_{ltz0}UEL_3 + HVG_{ltz1}vas\right) - vol_x$
WF_	WF_y	Al-geb	$KF\left(LVG_{ltz0}OEL + LVG_{ltz1}\left(HVG_{ltz0}UEL_3 + HVG_{ltz1}vas\right) - WF_x\right) - TFWF_y$
vf	vf	Ex-tAl-geb	$ue\left(-vf_0 + vout\right)$
Xad-Ifd	$XadIfd$	Ex-tAl-geb	0
a	a	Ex-tAl-geb	0
vbus	$vbus$	Ex-tAl-geb	0
vd	vd	Ex-tAl-	0

Services

Name	Symbol	Equation	Type
UEL0	$UEL0$	-999	ConstService
OEL0	$OEL0$	999	ConstService
ulim	$ulim$	9999	ConstService
llim	$llim$	-9999	ConstService
SG0	$SG0$	0	ConstService
zero	$zero$	0	ConstService
VA0	V_{A0}	$LR_{limzi}LR_x + LR_{limzl}zero - SGSWVOS_{s2} + vf_0$	PostInitService
vref0	V_{ref0}	$vref$	PostInitService
efdu	efd_u	$-KCXadIfd + VRMAX\sqrt{vd^2 + vq^2}$	VarService
efdl	efd_l	$VRMIN\sqrt{vd^2 + vq^2}$	VarService

Discretes

Name	Symbol	Type	Info
SWUEL	SW_{UEL}	Switcher	
SWVOS	SW_{VOS}	Switcher	
LR_lim	lim_{LR}	HardLimiter	
vil_lim	lim_{vil}	HardLimiter	
HVG1_lt	$None_{HVG1}$	LessThan	
LL_LT	LT_{LL}	LessThan	
LL1_LT	LT_{LL1}	LessThan	
LA_lim	lim_{LA}	AntiWindup	Limiter in Lag
HVG_lt	$None_{HVG}$	LessThan	
LVG_lt	$None_{LVG}$	LessThan	
vol_lim	lim_{vol}	HardLimiter	

Blocks

Name	Symbol	Type	Info
LG	LG	Lag	Voltage transducer
LR	LR	GainLimiter	Exciter output current gain limiter
vil	vil	GainLimiter	Exciter voltage input limiter
HVG1	$HVG1$	HVGate	HVGate after V_I
LL	LL	LeadLag	Lead-lag compensator
LL1	$LL1$	LeadLag	Lead-lag compensator 1
LA	LA	LagAntiWindup	V_A, Anti-windup lag
HVG	HVG	HVGate	HVGate for under excitation
LVG	LVG	LVGate	HVGate for over excitation
vol	vol	GainLimiter	Exciter output limiter
WF	WF	Washout	V_F, Stablizing circuit feedback

Setpoints

Name	Target	Info
vref	vref0	Initial reference voltage input

Config Fields in [ESST1A]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

ESAC5A

Exciter ESAC5A.

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
syn		Synchronous generator idx			mandatory
TR	T_R	Sensing Time Constant	0.010	*p.u*	
TA	T_A	Voltage Regulator Time Constant	0.040	*p.u*	
KA	K_A	Voltage Regulator Gain	80		
VRMIN	$V_R min$	V_R lower limit	-7.300	*p.u*	
VRMAX	$V_R max$	V_R upper limit	7.300	*p.u*	
TE	T_E	Integrator Time Constant	0.800	*p.u*	non_negative
KF	K_F	Feedback Gain	0.030		
TF1	T_{F1}	Lag Time Constant	1	*p.u*	
TF2	T_{F2}	Lead-Lag Time Constant (pole)	0.800	*p.u*	
TF3	T_{F3}	Lead-Lag Time Constant (zero)	1	*p.u*	
KE	K_E	Exciter Feedback Gain	1		
E1	E_1	First saturation point	0	*p.u.*	
SE1	S_E1	Value at first saturation point	0	*p.u.*	
E2	E_2	Second saturation point	1	*p.u.*	
SE2	S_E2	Value at second saturation point	1	*p.u.*	
ue	u_e	effective online status	0		
Sn	S_m	Rated power from generator	0	*MVA*	
Vn	V_m	Rated voltage from generator	0	*kV*	
bus	bus	Bus idx of the generators	0		

Variables

Name	Symbol	Type	Description	Unit	Properties
LP_y	LP_y	State	State in lag transfer function		v_str
VR_y	VR_y	State	State in lag TF		v_str
LL_x	LL_x	State	State in lead-lag		v_str
WF_x	WF_x	State	State in washout filter		v_str
INT_y	INT_y	State	Integrator output		v_str
omega	ω	ExtState	Generator speed		
v	v	Algeb	Input to exciter (bus v or Eterm)		v_str
vout	$vout$	Algeb	Exciter final output voltage		v_str
UEL	UEL	Algeb	Interface var for under exc. limiter		v_str
OEL	OEL	Algeb	Interface var for over exc. limiter		v_str
Vs	Vs	Algeb	Voltage compensation from PSS		v_str
vref	$vref$	Algeb	Reference voltage input	*p.u.*	v_str
vi	vi	Algeb	Total voltage input	*pu*	v_str
LL_y	LL_y	Algeb	Output of lead-lag		v_str
WF_y	WF_y	Algeb	Output of washout filter		v_str
Se	Se	Algeb	saturation output		v_str
VFE	VFE	Algeb	Combined saturation feedback	*p.u.*	v_str
vf	vf	ExtAlgeb	Excitation field voltage to generator		
XadIfd	$XadIfd$	ExtAlgeb	Armature excitation current		
a	a	ExtAlgeb	Bus voltage phase angle		
vbus	$vbus$	ExtAlgeb	Bus voltage magnitude		

Initialization Equations

Name	Symbol	Type	Initial Value
LP_y	LP_y	State	v
VR_y	VR_y	State	$KAvi$
LL_x	LL_x	State	VR_y
WF_x	WF_x	State	LL_y
INT_y	INT_y	State	vf_0
omega	ω	ExtState	
v	v	Algeb	$vbus$
vout	$vout$	Algeb	$uevf_0$
UEL	UEL	Algeb	UEL_0
OEL	OEL	Algeb	OEL_0
Vs	Vs	Algeb	0
vref	$vref$	Algeb	$v + \frac{VFE}{KA}$
vi	vi	Algeb	$ue\left(-v + vref\right)$
LL_y	LL_y	Algeb	VR_y
WF_y	WF_y	Algeb	0
Se	Se	Algeb	$SAT_B\left(INT_y - SAT_A\right)^2$ Indicator $\left(INT_y > SAT_A\right)$
VFE	VFE	Algeb	$INT_yKE + Se$
vf	vf	ExtAlgeb	
XadIfd	$XadIfd$	ExtAlgeb	
a	a	ExtAlgeb	
vbus	$vbus$	ExtAlgeb	

Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
LP_y	LP_y	State	$-LP_y + v$	T_R
VR_y	VR_y	State	$KAvi - VR_y$	T_A
LL_x	LL_x	State	$-LL_x + VR_y$	T_{F2}
WF_x	WF_x	State	$LL_y - WF_x$	T_{F1}
INT_y	INT_y	State	$ue\left(-VFE + VR_y\right)$	T_E
omega	ω	ExtState	0	

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
v	v	Algeb	$-v + vbus$
vout	$vout$	Algeb	$INT_y ue - vout$
UEL	UEL	Algeb	$-UEL + UEL_0$
OEL	OEL	Algeb	$-OEL + OEL_0$
Vs	Vs	Algeb	$-Vs$
vref	$vref$	Algeb	$-vref + vref_0$
vi	vi	Algeb	$ue\left(-LP_y + Vs - WF_y + vref\right) - vi$
LL_y	LL_y	Algeb	$LL_{LTz1}\left(LL_y - VR_y\right) + LL_x TF_2 - LL_y TF_2 + TF_3\left(-LL_x + VR_y\right)$
WF_y	WF_y	Algeb	$KF\left(LL_y - WF_x\right) - TF_1 WF_y$
Se	Se	Algeb	$ue\left(SAT_B SL_{z0}\left(INT_y - SAT_A\right)^2 - Se\right)$
VFE	VFE	Algeb	$ue\left(INT_y KE + Se - VFE\right)$
vf	vf	ExtAlgeb	$ue\left(-vf_0 + vout\right)$
XadIfd	$XadIfd$	ExtAlgeb	0
a	a	ExtAlgeb	0
vbus	$vbus$	ExtAlgeb	0

Services

Name	Symbol	Equation	Type
UEL0	$UEL0$	0	ConstService
OEL0	$OEL0$	0	ConstService
VRMAXu	$VRMAXu$	$VRMAXue - 999ue + 999$	ConstService
VRMINu	$VRMINu$	$VRMINue + 999ue - 999$	ConstService
SAT_E1	E_{SAT}^{1c}	E_1	ConstService
SAT_E2	E_{SAT}^{2c}	$E_2 - 2SAT_{zSE2} + 2$	ConstService
SAT_SE1	SE_{SAT}^{1c}	SE_1	ConstService
SAT_SE2	SE_{SAT}^{2c}	$-2SAT_{zSE2} + SE_2 + 2$	ConstService
SAT_a	a_{SAT}	$\sqrt{\frac{SAT_{E1}SAT_{SE1}}{SAT_{E2}SAT_{SE2}}}\left(\text{Indicator}\left(SAT_{SE2} > 0\right) + \text{Indicator}\left(SAT_{SE2}\right.\right.$	ConstService
SAT_A	A_{SAT}^q	$SAT_{E2} - \frac{SAT_{E1} - SAT_{E2}}{SAT_a - 1}$	ConstService
SAT_B	B_{SAT}^q	$\frac{SAT_{E2}SAT_{SE2}\left(SAT_a - 1\right)^2\left(\text{Indicator}\left(SAT_a > 0\right) + \text{Indicator}\left(SAT_a < 0\right)\right)}{\left(SAT_{E1} - SAT_{E2}\right)^2}$	ConstService
vref0	V_{ref0}	$vref$	PostInitService

Discretes

Name	Symbol	Type	Info
VR_lim	lim_{VR}	AntiWindup	Limiter in Lag
LL_LT	LT_{LL}	LessThan	
SL	SL	LessThan	

Blocks

Name	Symbol	Type	Info
LP	LP	Lag	Voltage transducer
VR	VR	LagAntiWindup	
LL	LL	LeadLag	
WF	WF	Washout	
SAT	SAT	ExcQuadSat	Field voltage saturation
INT	INT	Integrator	V_E, integrator

Setpoints

Name	Target	Info
vref	vref0	Initial reference voltage input

Config Fields in [ESAC5A]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

5.3.14 Experimental

Experimental group

Common Parameters: u, name

5.3.15 FreqMeasurement

Frequency measurements.

Common Parameters: u, name, bus

Common Variables: f

Available models: *BusFreq*, *BusROCOF*, *FreqDiv*

BusFreq

Bus frequency measurement. Outputs frequency in per unit value.

The bus frequency output variable is f. The frequency deviation variable is WO_y.

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
ue	u_e	effective online status	0		
bus		bus idx			mandatory
Tf	T_f	input digital filter time const	0.020	*sec*	non_negative
Tw	T_w	washout time const	0.100	*sec*	non_negative
fn	f_n	nominal frequency	60	*Hz*	non_negative

Variables

Name	Symbol	Type	Description	Unit	Properties
L_y	L_y	State	State in lag transfer function		v_str
WO_x	WO_x	State	State in washout filter		v_str
WO_y	WO_y	Algeb	frequency deviation	*p.u. (Hz)*	v_str
f	f	Algeb	frequency output	*p.u. (Hz)*	v_str
a	a	ExtAlgeb			
v	v	ExtAlgeb			

Initialization Equations

Name	Symbol	Type	Initial Value
L_y	L_y	State	$a - a_0$
WO_x	WO_x	State	L_y
WO_y	WO_y	Algeb	0
f	f	Algeb	1
a	a	ExtAlgeb	
v	v	ExtAlgeb	

Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
L_y	L_y	State	$-L_y + a - a_0$	T_f
WO_x	WO_x	State	$L_y - WO_x$	T_w

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
WO_y	WO_y	Algeb	$-TwWO_y + iwn\left(L_y - WO_x\right)$
f	f	Algeb	$WO_y - f + 1$
a	a	ExtAlgeb	0
v	v	ExtAlgeb	0

Services

Name	Symbol	Equation	Type
iwn	$1/\omega_n$	$\frac{u}{2\pi fn}$	ConstService

Blocks

Name	Symbol	Type	Info
L	L	Lag	digital filter
WO	WO	Washout	angle washout

Config Fields in [BusFreq]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

BusROCOF

Bus frequency and ROCOF measurement.

The ROCOF output variable is `Wf_y`.

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
ue	u_e	effective online status	0		
bus		bus idx			mandatory
Tf	T_f	input digital filter time const	0.020	*sec*	non_negative
Tw	T_w	washout time const	0.100	*sec*	non_negative
fn	f_n	nominal frequency	60	*Hz*	non_negative
Tr	T_r	frequency washout time constant	0.100		

Variables

Name	Symbol	Type	Description	Unit	Properties
L_y	L_y	State	State in lag transfer function		v_str
WO_x	WO_x	State	State in washout filter		v_str
Wf_x	Wf_x	State	State in washout filter		v_str
WO_y	WO_y	Algeb	frequency deviation	*p.u. (Hz)*	v_str
f	f	Algeb	frequency output	*p.u. (Hz)*	v_str
Wf_y	Wf_y	Algeb	Output of washout filter		v_str
a	a	ExtAlgeb			
v	v	ExtAlgeb			

Initialization Equations

Name	Symbol	Type	Initial Value
L_y	L_y	State	$a - a_0$
WO_x	WO_x	State	L_y
Wf_x	Wf_x	State	f
WO_y	WO_y	Algeb	0
f	f	Algeb	1
Wf_y	Wf_y	Algeb	0
a	a	ExtAlgeb	
v	v	ExtAlgeb	

Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
L_y	L_y	State	$-L_y + a - a_0$	T_f
WO_x	WO_x	State	$L_y - WO_x$	T_w
Wf_x	Wf_x	State	$-Wf_x + f$	T_r

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
WO_y	WO_y	Algeb	$-TwWO_y + iwn\left(L_y - WO_x\right)$
f	f	Algeb	$WO_y - f + 1$
Wf_y	Wf_y	Algeb	$-TrWf_y - Wf_x + f$
a	a	ExtAlgeb	0
v	v	ExtAlgeb	0

Services

Name	Symbol	Equation	Type
iwn	$1/\omega_n$	$\frac{u}{2\pi fn}$	ConstService

Blocks

Name	Symbol	Type	Info
L	L	Lag	digital filter
WO	WO	Washout	angle washout
Wf	Wf	Washout	frequency washout yielding ROCOF

Config Fields in [BusROCOF]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

FreqDiv

Frequency Divider model.

Estimates bus frequencies from generator rotor speeds using the network susceptance matrix, based on Milano & Ortega (2017).

Requires one device per bus. Typical usage:

```
ss = andes.load('case.json', setup=False)
for bus_idx in ss.Bus.idx.v:
    ss.add('FreqDiv', bus=bus_idx)
ss.setup()
```

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
bus		bus idx			mandatory
ue	u_e	effective online status	0		

Variables

Name	Symbol	Type	Description	Unit	Properties
f	f	Algeb	bus frequency		v_str

Initialization Equations

Name	Symbol	Type	Initial Value
f	f	Algeb	1

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
f	f	Algeb	0

Config Fields in [FreqDiv]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

5.3.16 Information

Group for information container models.

Available models: *Summary*

Summary

Class for storing system summary. Can be used for random information or notes.

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
field		field name			
comment		information, comment, or anything			
comment2		comment field 2			
comment3		comment field 3			
comment4		comment field 4			
ue	u_e	effective online status	0		

Config Fields in [Summary]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

5.3.17 Interface

Group for interface models.

Available models: *Fortescue*

Fortescue

Fortescue's symmetric component interface.

This model interfaces a positive-sequence, single-phase-equivalent bus with three buses representing three phases. It is effectively a transformer with one terminal on the primary side and three on the secondary. Only the positive sequence component on the secondary winding is used for simulation.

The positive-sequence voltage magnitude and angle of the secondary winding are named `vp` and `ap`.

The negative and zero sequence variables given in the d- and q-axis due the angle being undefined when the voltage is zero. The negative sequence voltages are `vnd` and `vnq` for the d- anx q-axis, respectively. Likewise, the zero-sequence voltages are `vzd` and `vzq`.

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
bus		bus idx for the single-phase equivalent			mandatory
busa		bus idx for phase a			mandatory
busb		bus idx for phase b			mandatory
busc		bus idx for phase c			mandatory
Sn	S_n	Power rating	100	*MW*	non_zero
r	r	resistance	0.001	*p.u.*	z
x	x	short-circuit reactance	0.025	*p.u.*	non_zero,z
g		iron loss	0	*p.u.*	y
b		magnetizing susceptance	0.005	*p.u.*	y
ue	u_e	effective online status	0		

Variables

Name	Symbol	Type	Description	Unit	Properties
vp	vp	Algeb	positive sequence voltage magnitude		v_str
ap	ap	Algeb	positive sequence voltage phase		v_str
vnd	vnd	Algeb	negative sequence voltage on d-axis (cos)		v_str
vnq	vnq	Algeb	negative sequence voltage on q-axis (sin)		v_str
vzd	vzd	Algeb	zero sequence voltage on d-axis (cos)		v_str
vzq	vzq	Algeb	zero sequence voltage on q-axis (sin)		v_str
a	a	ExtAlgeb	phase angle of single-phase eq. bus		
aa	aa	ExtAlgeb	phase angle of bus for phase a		
ab	ab	ExtAlgeb	phase angle of bus for phase b		
ac	ac	ExtAlgeb	phase angle of bus for phase c		
v	v	ExtAlgeb	voltage of single-phase eq. bus		
va	va	ExtAlgeb	voltage of bus for phase a		
vb	vb	ExtAlgeb	voltage of bus for phase b		
vc	vc	ExtAlgeb	voltage of bus for phase c		

Initialization Equations

Name	Symbol	Type	Initial Value
vp	vp	Algeb	$\frac{va}{3} + \frac{vb}{3} + \frac{vc}{3}$
ap	ap	Algeb	$aa + ab + ac$
vnd	vnd	Algeb	0.0
vnq	vnq	Algeb	0.0
vzd	vzd	Algeb	0.0
vzq	vzq	Algeb	0.0
a	a	ExtAlgeb	
aa	aa	ExtAlgeb	
ab	ab	ExtAlgeb	
ac	ac	ExtAlgeb	
v	v	ExtAlgeb	
va	va	ExtAlgeb	
vb	vb	ExtAlgeb	
vc	vc	ExtAlgeb	

Algebraic Equations

Nam	Sym-bol	Type	RHS of Equation "0 = g(x, y)"
vp	vp	Al-geb	$-vp+\frac{\sqrt{(vb\sin(-aa+ab+d_{120})-vc\sin(aa-ac+d_{120}))^2+(va+vb\cos(-aa+ab+d_{120})+vc\cos(aa-ac+d_{120}))^2}}{3}$
ap	ap	Al-geb	$aa-ap+\text{atan}_2\left(vb\sin(-aa+ab+d_{120})-vc\sin(aa-ac+d_{120}), va+vb\cos(-aa+ab+d_{120}\right.$
vnd	vnd	Al-geb	$va\cos(aa)+vb\cos(ab-d_{120})+vc\cos(ac+d_{120})-vnd$
vnq	vnq	Al-geb	$va\sin(aa)+vb\sin(ab-d_{120})+vc\sin(ac+d_{120})-vnq$
vzd	vzd	Al-geb	$va\cos(aa)+vb\cos(ab)+vc\cos(ac)-vzd$
vzq	vzq	Al-geb	$va\sin(aa)+vb\sin(ab)+vc\sin(ac)-vzq$
a	a	Ex-tAl-geb	$ue\left(v^2(g+ghk)-vvp(bhk\sin(a-ap)+ghk\cos(a-ap))\right)$
aa	aa	Ex-tAl-geb	$\frac{ue\left(-vva(-bhk\sin(a-aa)+ghk\cos(a-aa))+va^2(g+ghk)\right)}{3}$
ab	ab	Ex-tAl-geb	$\frac{ue\left(-vvb(bhk\sin(-a+ab+d_{120})+ghk\cos(-a+ab+d_{120}))+vb^2(g+ghk)\right)}{3}$
ac	ac	Ex-tAl-geb	$\frac{ue\left(-vvc(-bhk\sin(a-ac+d_{120})+ghk\cos(a-ac+d_{120}))+vc^2(g+ghk)\right)}{3}$
v	v	Ex-tAl-geb	$ue\left(-v^2(b+bhk)-vvp(-bhk\cos(a-ap)+ghk\sin(a-ap))\right)$
va	va	Ex-tAl-geb	$\frac{ue\left(vva(bhk\cos(a-aa)+ghk\sin(a-aa))-va^2(b+bhk)\right)}{3}$
vb	vb	Ex-tAl-geb	$\frac{ue\left(vvb(bhk\cos(-a+ab+d_{120})-ghk\sin(-a+ab+d_{120}))-vb^2(b+bhk)\right)}{3}$
vc	vc	Ex-tAl-geb	$\frac{ue\left(vvc(bhk\cos(a-ac+d_{120})+ghk\sin(a-ac+d_{120}))-vc^2(b+bhk)\right)}{3}$

Services

Name	Symbol	Equation	Type
yhk	y_{hk}	$\frac{u}{r+ix}$	ConstService
ghk	g_{hk}	re (yhk)	ConstService
bhk	b_{hk}	im (yhk)	ConstService
d120	120°	$\frac{2\pi}{3}$	ConstService

Config Fields in [Fortescue]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

5.3.18 Motor

Induction Motor group

Common Parameters: u, name, bus

Available models: *Motor3*, *Motor5*

Motor3

Third-order induction motor model.

See "Power System Modelling and Scripting" by F. Milano.

To simulate motor startup, set the motor status `u` to `0` and use a `Toggle` to control the model.

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
bus		interface bus id			mandatory
Sn	S_n	Power rating	100		
Vn	V_n	AC voltage rating	110		
fn	f	rated frequency	60		
rs	r_s	rotor resistance	0.010		non_zero,z
xs	x_s	rotor reactance	0.150		non_zero,z
rr1	r_{R1}	1st cage rotor resistance	0.050		non_zero,z
xr1	x_{R1}	1st cage rotor reactance	0.150		non_zero,z
rr2	r_{R2}	2st cage rotor resistance	0.001		non_zero,z
xr2	x_{R2}	2st cage rotor reactance	0.040		non_zero,z
xm	x_m	magnetization reactance	5		non_zero,z
Hm	H_m	Inertia constant	3	*kWs/KVA*	power
c1	c_1	1st coeff. of Tm(w)	0.100		
c2	c_2	2nd coeff. of Tm(w)	0.020		
c3	c_3	3rd coeff. of Tm(w)	0.020		
zb	z_b	Allow working as brake	1		
ue	u_e	effective online status	0		

Variables

Name	Symbol	Type	Description	Unit	Properties
slip	$slip$	State			v_str
e1d	$e1d$	State	real part of 1st cage voltage		v_str
e1q	$e1q$	State	imaginary part of 1st cage voltage		v_str
vd	vd	Algeb	d-axis voltage		
vq	vq	Algeb	q-axis voltage		
p	p	Algeb			v_str
q	q	Algeb			v_str
Id	Id	Algeb			v_str
Iq	Iq	Algeb			
te	te	Algeb			v_str
tm	tm	Algeb			v_str
a	a	ExtAlgeb	Bus voltage phase angle		
v	v	ExtAlgeb	Bus voltage magnitude		

Initialization Equations

Name	Symbol	Type	Initial Value
slip	$slip$	State	$1.0u$
e1d	$e1d$	State	$0.05u$
e1q	$e1q$	State	$0.9u$
vd	vd	Algeb	
vq	vq	Algeb	
p	p	Algeb	$u\left(Idvd + Iqvq\right)$
q	q	Algeb	$u\left(Idvq - Iqvd\right)$
Id	Id	Algeb	1
Iq	Iq	Algeb	
te	te	Algeb	$u\left(Ide1d + Iqe1q\right)$
tm	tm	Algeb	$u\left(aa + bbslip + c_2 slip^2\right)$
a	a	ExtAlgeb	
v	v	ExtAlgeb	

Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
slip	$slip$	State	$ue\left(-te + tm\right)$	M
e1d	$e1d$	State	$ue\left(e1qslipwb - \frac{Iq(x_0 - x_1) + e1d}{T_{10}}\right)$	
e1q	$e1q$	State	$ue\left(-e1dslipwb - \frac{-Id(x_0 - x_1) + e1q}{T_{10}}\right)$	

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
vd	vd	Algeb	$-uev\sin\left(a\right) - vd$
vq	vq	Algeb	$uev\cos\left(a\right) - vq$
p	p	Algeb	$-p + ue\left(Idvd + Iqvq\right)$
q	q	Algeb	$-q + ue\left(Idvq - Iqvd\right)$
Id	Id	Algeb	$ue\left(-Idrs + Iqx_1 - e1d + vd\right)$
Iq	Iq	Algeb	$ue\left(-Idx_1 - Iqrs - e1q + vq\right)$
te	te	Algeb	$-te + ue\left(Ide1d + Iqe1q\right)$
tm	tm	Algeb	$-tm + ue\left(aa + bbslip + c_2 slip^2\right)$
a	a	ExtAlgeb	p
v	v	ExtAlgeb	q

Services

Name	Symbol	Equation	Type
wb	ω_b	$2\pi fn$	ConstService
x0	x_0	$xm + xs$	ConstService
x1	x'	$\frac{xmxr_1}{xm+xr_1} + xs$	ConstService
T10	T'_0	$\frac{xm+xr_1}{rr_1 wb}$	ConstService
M	M	$2Hm$	ConstService
aa	α	$c_1 + c_2 + c_3$	ConstService
bb	β	$-c_2 - 2c_3$	ConstService

Config Fields in [Motor3]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

Motor5

Fifth-order induction motor model.

See "Power System Modelling and Scripting" by F. Milano.

To simulate motor startup, set the motor status `u` to `0` and use a `Toggle` to control the model.

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
bus		interface bus id			mandatory
Sn	S_n	Power rating	100		
Vn	V_n	AC voltage rating	110		
fn	f	rated frequency	60		
rs	r_s	rotor resistance	0.010		non_zero,z
xs	x_s	rotor reactance	0.150		non_zero,z
rr1	r_{R1}	1st cage rotor resistance	0.050		non_zero,z
xr1	x_{R1}	1st cage rotor reactance	0.150		non_zero,z
rr2	r_{R2}	2st cage rotor resistance	0.001		non_zero,z
xr2	x_{R2}	2st cage rotor reactance	0.040		non_zero,z
xm	x_m	magnetization reactance	5		non_zero,z
Hm	H_m	Inertia constant	3	*kWs/KVA*	power
c1	c_1	1st coeff. of Tm(w)	0.100		
c2	c_2	2nd coeff. of Tm(w)	0.020		
c3	c_3	3rd coeff. of Tm(w)	0.020		
zb	z_b	Allow working as brake	1		
ue	u_e	effective online status	0		

Variables

Name	Symbol	Type	Description	Unit	Properties
slip	$slip$	State			v_str
e1d	$e1d$	State	real part of 1st cage voltage		v_str
e1q	$e1q$	State	imaginary part of 1st cage voltage		v_str
e2d	$e2d$	State	real part of 2nd cage voltage		v_str
e2q	$e2q$	State	imag part of 2nd cage voltage		v_str
vd	vd	Algeb	d-axis voltage		
vq	vq	Algeb	q-axis voltage		
p	p	Algeb			v_str
q	q	Algeb			v_str
Id	Id	Algeb			v_str
Iq	Iq	Algeb			v_str
te	te	Algeb			v_str
tm	tm	Algeb			v_str
a	a	ExtAlgeb	Bus voltage phase angle		
v	v	ExtAlgeb	Bus voltage magnitude		

Initialization Equations

Name	Symbol	Type	Initial Value
slip	$slip$	State	$1.0u$
e1d	$e1d$	State	$0.05u$
e1q	$e1q$	State	$0.9u$
e2d	$e2d$	State	$0.05u$
e2q	$e2q$	State	$0.9u$
vd	vd	Algeb	
vq	vq	Algeb	
p	p	Algeb	$u\left(Idvd + Iqvq\right)$
q	q	Algeb	$u\left(Idvq - Iqvd\right)$
Id	Id	Algeb	$0.9u$
Iq	Iq	Algeb	$0.1u$
te	te	Algeb	$u\left(Ide2d + Iqe2q\right)$
tm	tm	Algeb	$u\left(aa + bbslip + c_2 slip^2\right)$
a	a	ExtAlgeb	
v	v	ExtAlgeb	

Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
slip	$slip$	State	$ue\left(-te + tm\right)$	M
e1d	$e1d$	State	$ue\left(e1qslipwb - \frac{Iq(x_0 - x_1) + e1d}{T_{10}}\right)$	
e1q	$e1q$	State	$ue\left(-e1dslipwb - \frac{-Id(x_0 - x_1) + e1q}{T_{10}}\right)$	
e2d	$e2d$	State	$ue\left(e1qslipwb - slipwb\left(e1q - e2q\right) + \frac{-Iq(x_1 - x_2) + e1d - e2d}{T_{20}} - \frac{Iq(x_0 -}{T}\right.$	
e2q	$e2q$	State	$ue\left(-e1dslipwb + slipwb\left(e1d - e2d\right) + \frac{Id(x_1 - x_2) + e1q - e2q}{T_{20}} - \frac{-Id(x}{}\right.$	

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
vd	vd	Algeb	$-uev\sin(a) - vd$
vq	vq	Algeb	$uev\cos(a) - vq$
p	p	Algeb	$-p + ue\,(Idvd + Iqvq)$
q	q	Algeb	$-q + ue\,(Idvq - Iqvd)$
Id	Id	Algeb	$ue\,(-Idrs + Iqx_2 - e2d + vd)$
Iq	Iq	Algeb	$ue\,(-Idx_2 - Iqrs - e2q + vq)$
te	te	Algeb	$-te + ue\,(Ide2d + Iqe2q)$
tm	tm	Algeb	$-tm + ue\left(aa + bbslip + c_2slip^2\right)$
a	a	ExtAlgeb	p
v	v	ExtAlgeb	q

Services

Name	Symbol	Equation	Type
wb	ω_b	$2\pi fn$	ConstService
x0	x_0	$xm + xs$	ConstService
x1	x'	$\frac{xmxr_1}{xm+xr_1} + xs$	ConstService
T10	T_0'	$\frac{xm+xr_1}{rr_1wb}$	ConstService
M	M	$2Hm$	ConstService
aa	α	$c_1 + c_2 + c_3$	ConstService
bb	β	$-c_2 - 2c_3$	ConstService
x2	x''	$\frac{xmxr_1xr_2}{xmxr_1+xmxr_2+xr_1xr_2} + xs$	ConstService
T20	T_0''	$\frac{\frac{xmxr_1}{xm+xr_1}+xr_2}{rr_2wb}$	ConstService

Config Fields in [Motor5]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

5.3.19 OutputSelect

Group for selecting outputs.

Available models: *Output*

Output

Model for specifying output models and/or variables.

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
model		Name of the model			mandatory
varname		Variable name			
dev		Device name			
ue	u_e	effective online status	0		

Config Fields in [Output]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

5.3.20 PLL

Phase-locked loop models.

Common Parameters: u, name, bus

Common Variables: am

Available models: *PLL1*, *PLL2*

PLL1

Simple Phasor Lock Loop (PLL) using one PI controller. The PI controller minimizes the error between the input and output angle.

Input bus angle signal -> Lag filter 1 with Tf -> Output angle *af_y*.

(af_y - am) -> PI Controller (Kp, Ki) -> PI_y

Estimated angle ae = (2 * pi * fn * PI_y) -> Lag filter 2 with Tp -> am.

The output signal is `am`, a state variable.

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
bus		bus idx			mandatory
fn	f_n	nominal frequency	60	*Hz*	
Kp	K_p	proportional gain	0.100		
Ki	K_i	integral gain	0.100		
Tf	T_f	input digital filter time const	0.050	*sec*	
Tp	T_p	output filter time const.	0.050	*sec*	
ue	u_e	effective online status	0		

Variables

Name	Symbol	Type	Description	Unit	Properties
af_y	af_y	State	State in lag transfer function		v_str
PI_xi	π_ξ	State	Integrator output		v_str
ae	ae	State	PLL angle output before filter		v_str
am	am	State	PLL output angle after filtering		v_str
PI_y	π_y	Algeb	PI output		v_str
a	a	ExtAlgeb	Bus voltage angle		

Initialization Equations

Name	Symbol	Type	Initial Value
af_y	af_y	State	a
PI_xi	π_ξ	State	0.0
ae	ae	State	a
am	am	State	a
PI_y	π_y	Algeb	$Kpu\left(af_y - am\right)$
a	a	ExtAlgeb	

Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
af_y	af_y	State	$a - af_y$	T_f
PI_xi	π_ξ	State	$Kiu\left(af_y - am\right)$	
ae	ae	State	$2\pi\pi_y fn$	
am	am	State	$ae - am$	T_p

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
PI_y	π_y	Algeb	$Kpu\left(af_y - am\right) + \pi_\xi - \pi_y$
a	a	ExtAlgeb	0

Blocks

Name	Symbol	Type	Info
af	af	Lag	input angle signal filter
PI	PI	PIController	PI controller

Config Fields in [PLL1]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

PLL2

Synchronously-rotating Reference Frame (SRF) Phasor Lock Loop (PLL).

The PLL minimizes `vq = v sin(a - am)` using a PI controller.

The output signal is `am`, a state variable.

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
bus		bus idx			mandatory
fn	f_n	nominal frequency	60	*Hz*	
Kp	K_p	proportional gain	0.100		
Ki	K_i	integral gain	0.100		
ue	u_e	effective online status	0		

Variables

Name	Symbol	Type	Description	Unit	Properties
PI_xi	π_ξ	State	Integrator output		v_str
am	am	State	PLL angle output		v_str
PI_y	π_y	Algeb	PI output		v_str
a	a	ExtAlgeb	Bus voltage angle		
v	v	ExtAlgeb	Bus voltage magnitude		

Initialization Equations

Name	Symbol	Type	Initial Value
PI_xi	π_ξ	State	0
am	am	State	a
PI_y	π_y	Algeb	$Kpv \sin{(a - am)}$
a	a	ExtAlgeb	
v	v	ExtAlgeb	

Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
PI_xi	π_ξ	State	$Kiv \sin(a - am)$	
am	am	State	$2\pi\pi_y fn$	

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
PI_y	π_y	Algeb	$Kpv \sin(a - am) + \pi_\xi - \pi_y$
a	a	ExtAlgeb	0
v	v	ExtAlgeb	0

Blocks

Name	Symbol	Type	Info
PI	PI	PIController	

Config Fields in [PLL2]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

5.3.21 PSS

Power system stabilizer group.

Common Parameters: u, name, avr

Common Variables: vsout

Available models: *IEEEST*, *ST2CUT*

IEEEST

IEEEST stabilizer model. Automatically adds frequency measurement devices if not provided.

Input signals (MODE):

1. Rotor speed deviation (p.u.),
2. Bus frequency deviation (p.u.) (*),
3. Generator P electrical in Gen MVABase (p.u.),
4. Generator accelerating power (p.u.),
5. Bus voltage (p.u.),
6. Derivative of p.u. bus voltage.

(*) Due to the frequency measurement implementation difference, mode 2 is likely to yield different results across software.

Note: Blocks are named *F1*, *F2*, *LL1*, *LL2* and *WO* in sequence. Two limiters are named *VLIM* and *OLIM* in sequence.

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
avr		Exciter idx			mandatory
MODE		Input signal			mandatory
busr		Optional remote bus idx			
busf		BusFreq idx for mode 2			
A1	A_1	filter time const. (pole)	1		
A2	A_2	filter time const. (pole)	1		
A3	A_3	filter time const. (pole)	1		
A4	A_4	filter time const. (pole)	1		
A5	A_5	filter time const. (zero)	1		
A6	A_6	filter time const. (zero)	1		
T1	T_1	first leadlag time const. (zero)	1		
T2	T_2	first leadlag time const. (pole)	1		
T3	T_3	second leadlag time const. (pole)	1		
T4	T_4	second leadlag time const. (pole)	1		
T5	T_5	washout time const. (zero)	1		
T6	T_6	washout time const. (pole)	1		
KS	K_S	Gain before washout	1		
LSMAX	L_{SMAX}	Max. output limit	0.300		
LSMIN	L_{SMIN}	Min. output limit	-0.300		
VCU	V_{CU}	Upper enabling bus voltage	999	*p.u.*	
VCL	V_{CL}	Upper enabling bus voltage	-999	*p.u.*	
ue	u_e	effective online status	0		
syn		Retrieved generator idx	0		
bus		Retrieved bus idx			
Sn	S_n	Generator power base	0		

Variables

Name	Symbol	Type	Description	Unit	Properties
F1_x	F_{1x}	State	State in 2nd order LPF		v_str
F1_y	F_{1y}	State	Output of 2nd order LPF		v_str
F2_x1	F_{2x1}	State	State #1 in 2nd order lead-lag		v_str
F2_x2	F_{2x2}	State	State #2 in 2nd order lead-lag		v_str
LL1_x	LL_{1x}	State	State in lead-lag		v_str
LL2_x	LL_{2x}	State	State in lead-lag		v_str
WO_x	WO_x	State	State in washout filter		v_str
omega	ω	ExtState	Generator speed	*p.u.*	
vsout	$vsout$	Algeb	PSS output voltage to exciter		
sig	sig	Algeb	Input signal		v_str
F2_y	F_{2y}	Algeb	Output of 2nd order lead-lag		v_str
LL1_y	LL_{1y}	Algeb	Output of lead-lag		v_str
LL2_y	LL_{2y}	Algeb	Output of lead-lag		v_str
WO_y	WO_y	Algeb	Output of washout filter		v_str
Vss	Vss	Algeb	Voltage output before output limiter		
tm	tm	ExtAlgeb	Generator mechanical input		
te	te	ExtAlgeb	Generator electrical output		
v	v	ExtAlgeb	Bus (or busr, if given) terminal voltage		
f	f	ExtAlgeb	Bus frequency		
vi	vi	ExtAlgeb	Exciter input voltage		

Initialization Equations

Name	Symbol	Type	Initial Value
F1_x	F_{1x}	State	0
F1_y	F_{1y}	State	sig
F2_x1	F_{2x1}	State	0
F2_x2	F_{2x2}	State	F_{1y}
LL1_x	LL_{1x}	State	F_{2y}
LL2_x	LL_{2x}	State	LL_{1y}
WO_x	WO_x	State	$KSLL_{2y}$
omega	ω	ExtState	
vsout	$vsout$	Algeb	
sig	sig	Algeb	$ue\left(SW_{s1}\left(\omega-1\right)+\frac{SW_{s3}tm_0}{SnSb}+SW_{s4}\left(tm-tm_0\right)+SW_{s5}v\right)$
F2_y	F_{2y}	Algeb	F_{1y}
LL1_y	LL_{1y}	Algeb	F_{2y}
LL2_y	LL_{2y}	Algeb	LL_{1y}
WO_y	WO_y	Algeb	$WO_{LTz1}WO_x$
Vss	Vss	Algeb	
tm	tm	ExtAlgeb	
te	te	ExtAlgeb	
v	v	ExtAlgeb	
f	f	ExtAlgeb	
vi	vi	ExtAlgeb	

Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
F1_x	F_{1x}	State	$-A_1F_{1x}-F_{1y}+sig$	A_2
F1_y	F_{1y}	State	F_{1x}	
F2_x1	F_{2x1}	State	$-A_3F_{2x1}+F_{1y}-F_{2x2}$	A_4
F2_x2	F_{2x2}	State	F_{2x1}	
LL1_x	LL_{1x}	State	$F_{2y}-LL_{1x}$	T_2
LL2_x	LL_{2x}	State	$LL_{1y}-LL_{2x}$	T_4
WO_x	WO_x	State	$KSLL_{2y}-WO_x$	T_6
omega	ω	ExtState	0	

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
vsout	$vsout$	Algeb	$OLIM_{zi}Vssue - vsout$
sig	sig	Algeb	$-sig+ue\left(SW_{s1}\left(\omega-1\right)+SW_{s2}\left(f-1\right)+\frac{SW_{s3}te}{SnSb}+SW_{s4}\left(tm-tm_0\right)+SW_{s5}v+SW_{s6}d$
F2_y	F_{2y}	Algeb	$A_4A_5F_{2x1} + A_4F_{2x2} - A_4F_{2y} + A_6\left(-A_3F_{2x1}+F_{1y}-F_{2x2}\right) + F_{2LTz1}\left(-F_{1y}+F_{2y}\right)$
LL1_y	LL_{1y}	Algeb	$LL_{1LTz1}\left(-F_{2y}+LL_{1y}\right)+LL_{1x}T_2-LL_{1y}T_2+T_1\left(F_{2y}-LL_{1x}\right)$
LL2_y	LL_{2y}	Algeb	$LL_{2LTz1}\left(-LL_{1y}+LL_{2y}\right)+LL_{2x}T_4-LL_{2y}T_4+T_3\left(LL_{1y}-LL_{2x}\right)$
WO_y	WO_y	Algeb	$T_5WO_{LTz0}\left(KSLL_{2y}-WO_x\right)+T_6WO_{LTz1}WO_x-T_6WO_y$
Vss	Vss	Algeb	$LSMAXVLIM_{zu}+LSMINVLIM_{zl}+VLIM_{zi}WO_y-Vss$
tm	tm	ExtAlgeb	0
te	te	ExtAlgeb	0
v	v	ExtAlgeb	0
f	f	ExtAlgeb	0
vi	vi	ExtAlgeb	$uevsout$

Discretes

Name	Symbol	Type	Info
dv	dV/dt	Derivative	Finite difference of bus voltage
SW	SW	Switcher	
F2_LT	LT_{F2}	LessThan	
LL1_LT	LT_{LL1}	LessThan	
LL2_LT	LT_{LL2}	LessThan	
WO_LT	LT_{WO}	LessThan	
VLIM	$VLIM$	Limiter	Vss limiter
OLIM	$OLIM$	Limiter	output limiter

Blocks

Name	Symbol	Type	Info
F1	$F1$	Lag2ndOrd	
F2	$F2$	LeadLag2ndOrd	
LL1	$LL1$	LeadLag	
LL2	$LL2$	LeadLag	
Vks	Vks	Gain	
WO	WO	WashoutOrLag	

Config Fields in [IEEEST]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)
freq_model		BusFreq	default freq. measurement model	('BusFreq',)

ST2CUT

ST2CUT stabilizer model. Automatically adds frequency measurement devices if not provided.

Input signals (MODE and MODE2):

0 - Disable input signal 1 (s1) - Rotor speed deviation (p.u.), 2 (s2) - Bus frequency deviation (*) (p.u.), 3 (s3) - Generator P electrical in Gen MVABase (p.u.), 4 (s4) - Generator accelerating power (p.u.), 5 (s5) - Bus voltage (p.u.), 6 (s6) - Derivative of p.u. bus voltage.

(*) Due to the frequency measurement implementation difference, mode 2 is likely to yield different results across software.

Blocks are named *LL1*, *LL2*, *LL3*, *LL4* in sequence. Two limiters are named *VSS_lim* and *OLIM* in sequence.

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
avr		Exciter idx			mandatory
MODE		Input signal 1			mandatory
busr		Remote bus 1			

continues on next page

Table 9 – continued from previous page

Name	Symbol	Description	Default	Unit	Properties
busf		BusFreq idx for signal 1 mode 2			
MODE2		Input signal 2			
busr2		Remote bus 2			
busf2		BusFreq idx for signal 2 mode 2			
K1	K_1	Transducer 1 gain	1		
K2	K_2	Transducer 2 gain	1		
T1	T_1	Transducer 1 time const.	1		
T2	T_2	Transducer 2 time const.	1		
T3	T_3	Washout int. time const.	1		
T4	T_4	Washout delay time const.	0.200		
T5	T_5	Leadlag 1 time const. (1)	1		
T6	T_6	Leadlag 1 time const. (2)	0.500		
T7	T_7	Leadlag 2 time const. (1)	1		
T8	T_8	Leadlag 2 time const. (2)	1		
T9	T_9	Leadlag 3 time const. (1)	1		
T10	T_{10}	Leadlag 3 time const. (2)	0.200		
LSMAX	L_{SMAX}	Max. output limit	0.300		
LSMIN	L_{SMIN}	Min. output limit	-0.300		
VCU	V_{CU}	Upper enabling bus voltage	999	*p.u.*	
VCL	V_{CL}	Upper enabling bus voltage	-999	*p.u.*	
ue	u_e	effective online status	0		
syn		Retrieved generator idx	0		
bus		Retrieved bus idx			
Sn	S_n	Generator power base	0		

Variables

Name	Symbol	Type	Description	Unit	Properties
L1_y	L_{1y}	State	State in lag transfer function		v_str
L2_y	L_{2y}	State	State in lag transfer function		v_str
WO_x	WO_x	State	State in washout filter		v_str
LL1_x	LL_{1x}	State	State in lead-lag		v_str
LL2_x	LL_{2x}	State	State in lead-lag		v_str
LL3_x	LL_{3x}	State	State in lead-lag		v_str
omega	ω	ExtState	Generator speed	*p.u.*	
vsout	$vsout$	Algeb	PSS output voltage to exciter		
sig	sig	Algeb	Input signal		v_str
sig2	sig_2	Algeb	Input signal 2		v_str
IN	IN	Algeb	Sum of inputs		v_str
WO_y	WO_y	Algeb	Output of washout filter		v_str
LL1_y	LL_{1y}	Algeb	Output of lead-lag		v_str
LL2_y	LL_{2y}	Algeb	Output of lead-lag		v_str
LL3_y	LL_{3y}	Algeb	Output of lead-lag		v_str
VSS_x	VSS_x	Algeb	Value before limiter		v_str
tm	tm	ExtAlgeb	Generator mechanical input		
te	te	ExtAlgeb	Generator electrical output		
v	v	ExtAlgeb	Bus (or busr, if given) terminal voltage		
f	f	ExtAlgeb	Bus frequency		
vi	vi	ExtAlgeb	Exciter input voltage		
v2	v_2	ExtAlgeb	Bus (or busr2, if given) terminal voltage		
f2	f_2	ExtAlgeb	Bus frequency 2		

Initialization Equations

Name	Symbol	Type	Initial Value
L1_y	L_{1y}	State	$K_1 sig$
L2_y	L_{2y}	State	$K_2 sig_2$
WO_x	WO_x	State	IN
LL1_x	LL_{1x}	State	WO_y
LL2_x	LL_{2x}	State	LL_{1y}
LL3_x	LL_{3x}	State	LL_{2y}
omega	ω	ExtState	
vsout	$vsout$	Algeb	
sig	sig	Algeb	$SW_{s1}\left(\omega - 1\right) + \frac{SW_{s3} tm_0}{SnSb} + SW_{s4}\left(tm - tm_0\right) + SW_{s5} v$
sig2	sig_2	Algeb	$SW_{2s1}\left(\omega - 1\right) + \frac{SW_{2s3} tm_0}{SnSb} + SW_{2s4}\left(tm - tm_0\right) + SW_{2s5} v_2$
IN	IN	Algeb	$ue\left(L_{1y} + L_{2y}\right)$
WO_y	WO_y	Algeb	$WO_{LTz1} WO_x$
LL1_y	LL_{1y}	Algeb	WO_y
LL2_y	LL_{2y}	Algeb	LL_{1y}
LL3_y	LL_{3y}	Algeb	LL_{2y}
VSS_x	VSS_x	Algeb	LL_{3y}
tm	tm	ExtAlgeb	
te	te	ExtAlgeb	
v	v	ExtAlgeb	
f	f	ExtAlgeb	
vi	vi	ExtAlgeb	
v2	v_2	ExtAlgeb	
f2	f_2	ExtAlgeb	

Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
L1_y	L_{1y}	State	$K_1 sig - L_{1y}$	T_1
L2_y	L_{2y}	State	$K_2 sig_2 - L_{2y}$	T_2
WO_x	WO_x	State	$IN - WO_x$	T_4
LL1_x	LL_{1x}	State	$-LL_{1x} + WO_y$	T_6
LL2_x	LL_{2x}	State	$LL_{1y} - LL_{2x}$	T_8
LL3_x	LL_{3x}	State	$LL_{2y} - LL_{3x}$	T_{10}
omega	ω	ExtState	0	

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
vsout	$vsout$	Algeb	$OLIM_{zi}ue\left(LSMAXVSS_{limzu} + LSMINVSS_{limzl} + VSS_{limzi}VSS_x\right) - vsout$
sig	sig	Algeb	$SW_{s1}\left(\omega - 1\right) + SW_{s2}\left(f - 1\right) + \frac{SW_{s3}te}{SnSb} + SW_{s4}\left(tm - tm_0\right) + SW_{s5}v + SW_{s6}dv_v - sig$
sig2	sig_2	Algeb	$SW_{2s1}\left(\omega - 1\right) + SW_{2s2}\left(f_2 - 1\right) + \frac{SW_{2s3}te}{SnSb} + SW_{2s4}\left(tm - tm_0\right) + SW_{2s5}v_2 + SW_{2s6}dv_{2v} - sig_2$
IN	IN	Algeb	$-IN + ue\left(L_{1y} + L_{2y}\right)$
WO_y	WO_y	Algeb	$T_3WO_{LTz0}\left(IN - WO_x\right) + T_4WO_{LTz1}WO_x - T_4WO_y$
LL1_y	LL_{1y}	Algeb	$LL_{1LTz1}\left(LL_{1y} - WO_y\right) + LL_{1x}T_6 - LL_{1y}T_6 + T_5\left(-LL_{1x} + WO_y\right)$
LL2_y	LL_{2y}	Algeb	$LL_{2LTz1}\left(-LL_{1y} + LL_{2y}\right) + LL_{2x}T_8 - LL_{2y}T_8 + T_7\left(LL_{1y} - LL_{2x}\right)$
LL3_y	LL_{3y}	Algeb	$LL_{3LTz1}\left(-LL_{2y} + LL_{3y}\right) + LL_{3x}T_{10} - LL_{3y}T_{10} + T_9\left(LL_{2y} - LL_{3x}\right)$
VSS_x	VSS_x	Algeb	$LL_{3y} - VSS_x$
tm	tm	ExtAlgeb	0
te	te	ExtAlgeb	0
v	v	ExtAlgeb	0
f	f	ExtAlgeb	0
vi	vi	ExtAlgeb	$uevsout$
v2	v_2	ExtAlgeb	0
f2	f_2	ExtAlgeb	0

Services

Name	Symbol	Equation	Type
VOU	VOU	$VCUr + v_0$	ConstService
VOL	VOL	$VCLr + v_0$	ConstService

Discretes

Name	Symbol	Type	Info
dv	dv	Derivative	
dv2	$dv2$	Derivative	
SW	SW	Switcher	
SW2	$SW2$	Switcher	
WO_LT	LT_{WO}	LessThan	
LL1_LT	LT_{LL1}	LessThan	
LL2_LT	LT_{LL2}	LessThan	
LL3_LT	LT_{LL3}	LessThan	
VSS_lim	lim_{VSS}	HardLimiter	
OLIM	$OLIM$	Limiter	output limiter

Blocks

Name	Symbol	Type	Info
L1	$L1$	Lag	Transducer 1
L2	$L2$	Lag	Transducer 2
WO	WO	WashoutOrLag	
LL1	$LL1$	LeadLag	
LL2	$LL2$	LeadLag	
LL3	$LL3$	LeadLag	
VSS	VSS	GainLimiter	

Config Fields in [ST2CUT]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)
freq_model		BusFreq	default freq. measurement model	('BusFreq',)

5.3.22 PhasorMeasurement

Phasor measurements

Common Parameters: u, name, bus

Common Variables: am, vm

Available models: *PMU*

PMU

Simple phasor measurement unit model.

This model tracks the bus voltage magnitude and phase angle, each using a low-pass filter.

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
bus		bus idx			mandatory
Ta	T_a	angle filter time constant	0.100		
Tv	T_v	voltage filter time constant	0.100		
ue	u_e	effective online status	0		

Variables

Name	Symbol	Type	Description	Unit	Properties
am	am	State	phase angle measurement	*rad.*	v_str
vm	vm	State	voltage magnitude measurement	*p.u.(kV)*	v_str
a	a	ExtAlgeb	Bus voltage phase angle		
v	v	ExtAlgeb	Bus voltage magnitude		

Initialization Equations

Name	Symbol	Type	Initial Value
am	am	State	a
vm	vm	State	v
a	a	ExtAlgeb	
v	v	ExtAlgeb	

Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
am	am	State	$a - am$	T_a
vm	vm	State	$v - vm$	T_v

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
a	a	ExtAlgeb	0
v	v	ExtAlgeb	0

Config Fields in [PMU]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

5.3.23 RenAerodynamics

Renewable aerodynamics group.

Common Parameters: u, name, rego

Common Variables: theta

Available models: *WTARA1*, *WTARV1*

WTARA1

Wind turbine aerodynamics model (no wind speed details).

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
rego		Renewable governor idx			mandatory
Ka	K_a	Aerodynamics gain	1	*p.u./deg.*	non_negative
theta0	θ_0	Initial pitch angle	0	*deg.*	
ue	u_e	effective online status	0		

Variables

Name	Symbol	Type	Description	Unit	Properties
theta	θ	Algeb	Pitch angle	*rad*	v_str
Pmg	Pmg	ExtAlgeb			

Initialization Equations

Name	Symbol	Type	Initial Value
theta	θ	Algeb	$theta0r$
Pmg	Pmg	ExtAlgeb	

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
theta	θ	Algeb	$-\theta + theta0r$
Pmg	Pmg	ExtAlgeb	$-Ka\theta\left(\theta - \theta_0\right)$

Services

Name	Symbol	Equation	Type
theta0r	θ_{0r}	$\frac{\pi\theta_0}{180}$	ConstService

Config Fields in [WTARA1]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

WTARV1

Wind turbine aerodynamics model with wind velocity details.

Work is in progress.

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
rego		Renewable governor idx			mandatory
nblade		number of blades	3		
ngen		number of wind generator units	50		
npole		number of poles in generator	4		
R		rotor radius	30	*m*	
ngb		gear box ratio	5		
rho		air density	1.200	*kg/m3*	
ue	u_e	effective online status	0		
Sn	S_n		0		

Variables

Name	Symbol	Type	Description	Unit	Properties
theta	θ	Algeb	Pitch angle	*rad*	
Pmg	Pmg	ExtAlgeb			

Initialization Equations

Name	Symbol	Type	Initial Value
theta	θ	Algeb	
Pmg	Pmg	ExtAlgeb	

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
theta	θ	Algeb	0
Pmg	Pmg	ExtAlgeb	0

Config Fields in [WTARV1]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

5.3.24 RenExciter

Renewable electrical control (exciter) group.

Common Parameters: u, name, reg

Common Variables: Pref, Qref, wg, Pord

Available models: *REECA1*, *REECA1E*, *REECA1G*, *REECB1*

REECA1

Renewable energy electrical control.

There are two user-defined voltages: *Vref0* and *Vref1*.

- The difference between the initial bus voltage and *Vref0* should be within the voltage deadbands *dbd1* and *dbd2*.
- If *VFLAG=0*, the input to the second PI controller will be *Vref1*.

Regarding the additional reactive current injection during voltage dip:

- Exercise caution when coordinating the parameters `dbd1`, `dbd2`, `Vdip`, and `Vup` to avoid unintended responses.
- `Kqv` in `pu current / pu voltage deviation` controls the intensity of reactive power injection. The parameter needs to tuned properly to avoid voltage overshoot.
- When multiple renewable generators are connected to the same bus, `Kqv` shall be reduced accordingly to avoid excessive reactive power injection.

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
reg		Renewable generator idx			mandatory
busr		Optional remote bus for voltage control			
PFFLAG		Power factor control flag; 1-PF control, 0-Q control		*bool*	mandatory
VFLAG		Voltage control flag; 1-Q control, 0-V control		*bool*	mandatory
QFLAG		Q control flag; 1-V or Q control, 0-const. PF or Q		*bool*	mandatory
PFLAG		P speed-dependency flag; 1-has speed dep., 0-no dep.		*bool*	mandatory
PQFLAG		P/Q priority flag for I limit; 0-Q priority, 1-P priority		*bool*	mandatory
Vdip	V_{dip}	Low V threshold to activate Iqinj logic	0.800	*p.u.*	
Vup	V_{up}	V threshold above which to activate Iqinj logic	1.200	*p.u.*	
Trv	T_{rv}	Voltage filter time constant	0.020		
dbd1	d_{bd1}	Lower bound of the voltage deadband (<=0)	-0.020		
dbd2	d_{bd2}	Upper bound of the voltage deadband (>=0)	0.020		
Kqv	K_{qv}	Gain to compute Iqinj from V error (caution!!)	1		
Iqh1	I_{qh1}	Upper limit on Iqinj	999		
Iql1	I_{ql1}	Lower limit on Iqinj	-999		
Vref0	V_{ref0}	User defined Vref (if 0, use initial bus V)	1		
Iqfrz	I_{qfrz}	Hold Iqinj at the value for Thld (>0) seconds following a Vdip	0		
Thld	T_{hld}	Time for which Iqinj is held. Hold at Iqinj if>0; hold at State 1 if<0	0	*s*	
Thld2	T_{hld2}	Time for which IPMAX is held after voltage dip ends	0	*s*	
Tp	T_p	Filter time constant for Pe	0.020	*s*	

continues on next page

Table 10 – continued from previous page

Name	Symbol	Description	Default	Unit	Properties
QMax	Q_{max}	Upper limit for reactive power regulator	999		power
QMin	Q_{min}	Lower limit for reactive power regulator	-999		power
VMAX	V_{max}	Upper limit for voltage control	999		
VMIN	V_{min}	Lower limit for voltage control	-999		
Kqp	K_{qp}	Proportional gain for reactive power error	1		
Kqi	K_{qi}	Integral gain for reactive power error	0.100		
Kvp	K_{vp}	Proportional gain for voltage error	1		
Kvi	K_{vi}	Integral gain for voltage error	0.100		
Vref1	V_{ref1}	Voltage ref. if VFLAG=0	1		non_zero
Tiq	T_{iq}	Filter time constant for Iq	0.020		
dPmax	d_{Pmax}	Power reference max. ramp rate (>0)	999		power
dPmin	d_{Pin}	Power reference min. ramp rate (<0)	-999		power
PMAX	P_{max}	Max. active power limit > 0	999		power
PMIN	P_{min}	Min. active power limit	0		power
Imax	I_{max}	Max. apparent current limit	999		current
Tpord	T_{pord}	Filter time constant for power setpoint	0.020		
Vq1	V_{q1}	Reactive power V-I pair (point 1), voltage	0.200		
Iq1	I_{q1}	Reactive power V-I pair (point 1), current	2		current
Vq2	V_{q2}	Reactive power V-I pair (point 2), voltage	0.400		
Iq2	I_{q2}	Reactive power V-I pair (point 2), current	4		current
Vq3	V_{q3}	Reactive power V-I pair (point 3), voltage	0.800		
Iq3	I_{q3}	Reactive power V-I pair (point 3), current	8		current
Vq4	V_{q4}	Reactive power V-I pair (point 4), voltage	1		
Iq4	I_{q4}	Reactive power V-I pair (point 4), current	10		current
Vp1	V_{p1}	Active power V-I pair (point 1), voltage	0.200		
Ip1	I_{p1}	Active power V-I pair (point 1), current	2		current
Vp2	V_{p2}	Active power V-I pair (point 2), voltage	0.400		
Ip2	I_{p2}	Active power V-I pair (point 2), current	4		current
Vp3	V_{p3}	Active power V-I pair (point 3), voltage	0.800		
Ip3	I_{p3}	Active power V-I pair (point 3), current	8		current
Vp4	V_{p4}	Active power V-I pair (point 4), voltage	1		
Ip4	I_{p4}	Active power V-I pair (point 4), current	12		current
ue	u_e	effective online status	0		
bus		Retrieved bus idx			
gen		Retrieved StaticGen idx			
Sn	S_n		0		

Variables

Name	Symbol	Type	Description	Unit	Properties
s0_y	s_{0y}	State	State in lag transfer function		v_str
S1_y	S_{1y}	State	State in lag transfer function		v_str
PIQ_xi	PIQ_{ξ}	State	Integrator output		v_str
s4_y	s_{4y}	State	State in lag transfer function		v_str
pfilt_y	$pfilt_y$	State	State in lag TF		v_str
s5_y	s_{5y}	State	State in lag TF		v_str
PIV_xi	PIV_{ξ}	State	Integrator output		v_str
Pord	$Pord$	AliasState	Alias of s5_y		
vp	vp	Algeb	Sensed lower-capped voltage		v_str
pfaref	$pfaref$	Algeb	power factor angle ref	*rad*	v_str
Qref	$Qref$	Algeb	external Q ref	*p.u.*	v_str
Qcpf	$Qcpf$	Algeb	Q calculated from P and power factor	*p.u.*	v_str
PFsel	$PFsel$	Algeb	Output of PFFLAG selector		v_str
Qerr	$Qerr$	Algeb	Reactive power error		v_str
PIQ_ys	PIQ_{ys}	Algeb	PI summation before limit		v_str
PIQ_y	PIQ_y	Algeb	PI output		v_str
Vsel_x	$Vsel_x$	Algeb	Value before limiter		v_str
Verr	$Verr$	Algeb	Voltage error (Vref0)		v_str
dbV_y	dbV_y	Algeb	Deadband type 1 output		v_str
Iqinj	$Iqinj$	Algeb	Additional Iq signal during under- or over-voltage		v_str
wg	wg	Algeb	Drive train generator speed		v_str
Pref	$Pref$	Algeb	external P ref	*p.u.*	v_str
Psel	$Psel$	Algeb	Output selection of PFLAG		v_str
VDL1_y	VDL_{1y}	Algeb	Output of piecewise		v_str
VDL2_y	VDL_{2y}	Algeb	Output of piecewise		v_str
Ipmax	$Ipmax$	Algeb	Upper limit on Ipcmd		v_str
Iqmax	$Iqmax$	Algeb	Upper limit on Iqcmd		v_str
PIV_ys	PIV_{ys}	Algeb	PI summation before limit		v_str
PIV_y	PIV_y	Algeb	PI output		v_str
Qsel	$Qsel$	Algeb	Selection output of QFLAG		v_str
IpHL_x	$IpHL_x$	Algeb	Value before limiter		v_str
IqHL_x	$IqHL_x$	Algeb	Value before limiter		v_str
a	a	ExtAlgeb	Bus voltage angle		
v	v	ExtAlgeb	d-axis bus voltage magnitude		
Pe	Pe	ExtAlgeb	Retrieved Pe of RenGen		
Qe	Qe	ExtAlgeb	Retrieved Qe of RenGen		
Ipcmd	$Ipcmd$	ExtAlgeb	Retrieved Ipcmd of RenGen		
Iqcmd	$Iqcmd$	ExtAlgeb	Retrieved Iqcmd of RenGen		

Initialization Equations

Name	Symbol	Type	Initial Value
s0_y	s_{0y}	State	v
S1_y	S_{1y}	State	Pe
PIQ_xi	PIQ_{ξ}	State	0.0
s4_y	s_{4y}	State	$\frac{PFsel}{vp}$
pfilt_y	$pfilt_y$	State	$Pref$
s5_y	s_{5y}	State	$Psel$
PIV_xi	PIV_{ξ}	State	$-Iqcmd_0 SWQ_{s1}$
Pord	$Pord$	AliasState	
vp	vp	Algeb	$VLower_{zi} v + 0.01 VLower_{zl}$
pfaref	$pfaref$	Algeb	$pfaref_0$
Qref	$Qref$	Algeb	$-qref_0$
Qcpf	$Qcpf$	Algeb	q_0
PFsel	$PFsel$	Algeb	$QcpfSWPF_{s1} + QrefSWPF_{s0}$
Qerr	$Qerr$	Algeb	$PFlim_{zi} PFsel + PFlim_{zl} QMin + PFlim_{zu} QMax - Qe$
PIQ_ys	PIQ_{ys}	Algeb	$KqpQerrSWV_{s1}$
PIQ_y	PIQ_y	Algeb	$PIQ_{limzi} PIQ_{ys} + PIQ_{limzl} VMIN + PIQ_{limzu} VMAX$
Vsel_x	$Vsel_x$	Algeb	$PIQ_y SWV_{s1} + SWV_{s0} (QcpfSWPF_{s1} + QrefSWPF_{s0} + Vref_1)$
Verr	$Verr$	Algeb	$Vref0r - s_{0y}$
dbV_y	dbV_y	Algeb	$1.0 dbV_{dbzl} (Verr - dbd_1) + 1.0 dbV_{dbzu} (Verr - dbd_2)$
Iqinj	$Iqinj$	Algeb	$KqvVolt_{dip} dbV_y + fThld (1 - Volt_{dip}) (IqfrzpThld + KqvdbV_y nThld)$
wg	wg	Algeb	1.0
Pref	$Pref$	Algeb	$\frac{p_0}{wg}$
Psel	$Psel$	Algeb	$SWP_{s0} pfilt_y + SWP_{s1} pfilt_y wg$
VDL1_y	VDL_{1y}	Algeb	FixPiecewise $((Iq_1,\ Vq_1 \geq s_{0y}), (Iq_1 + kVq_{12} (-Vq_1 + s_{0y}),\ Vq_2 \geq s_{0y}), (Iq_2 + k$
VDL2_y	VDL_{2y}	Algeb	FixPiecewise $((Ip_1,\ Vp_1 \geq s_{0y}), (Ip_1 + kVp_{12} (-Vp_1 + s_{0y}),\ Vp_2 \geq s_{0y}), (Ip_2 +$
Ipmax	$Ipmax$	Algeb	$(1 - fThld_2) \left(\sqrt{Ipmax2sq0} SWPQ_{s0} + SWPQ_{s1} (zVDL_2 (Imaxr (1 - VDL2c)\right.$
Iqmax	$Iqmax$	Algeb	$\sqrt{Iqmax2sq0} SWPQ_{s1} + SWPQ_{s0} (zVDL_1 (Imaxr (1 - VDL1c) + VDL_{1y} VDI$
PIV_ys	PIV_{ys}	Algeb	$-Iqcmd_0 SWQ_{s1} + KvpSWQ_{s1} (-SWV_{s0} s_{0y} + VMAXVsel_{limzu} + VMINVse$
PIV_y	PIV_y	Algeb	$IqmaxPIV_{limzu} + IqminPIV_{limzl} + PIV_{limzi} PIV_{ys}$
Qsel	$Qsel$	Algeb	$PIV_y SWQ_{s1} + SWQ_{s0} s_{4y}$
IpHL_x	$IpHL_x$	Algeb	$\frac{s_{5y}}{vp}$
IqHL_x	$IqHL_x$	Algeb	$Iqinj + Qsel$
a	a	ExtAlgeb	
v	v	ExtAlgeb	
Pe	Pe	ExtAlgeb	
Qe	Qe	ExtAlgeb	
Ipcmd	$Ipcmd$	ExtAlgeb	
Iqcmd	$Iqcmd$	ExtAlgeb	

Differential Equations

Nam	Sym-bol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
s0_y	s_{0y}	State	$-s_{0y} + v$	T_{rv}
S1_y	S_{1y}	State	$Pe - S_{1y}$	T_p
PIQ_	PIQ_ξ	State	$Kqi\left(1 - Volt_{dip}\right)\left(2PIQ_y - 2PIQ_{ys} + QerrSWV_{s1}\right)$	
s4_y	s_{4y}	State	$\left(1 - Volt_{dip}\right)\left(\frac{PFsel}{vp} - s_{4y}\right)$	T_{iq}
pfilt_	$pfilt_y$	State	$Pref - pfilt_y$	0.02
s5_y	s_{5y}	State	$\left(1 - Volt_{dip}\right)\left(Psel - s_{5y}\right)$	T_{pord}
PIV_	PIV_ξ	State	$Kvi\left(1 - Volt_{dip}\right)\left(2PIV_y - 2PIV_{ys} + SWQ_{s1}\left(-SWV_{s0}s_{0y} + VMAX\right.\right.$	$_{nzu} + VMINV$
Pord	$Pord$	AliasSt	0	

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
vp	vp	Algeb	$VLower_{zi}v + 0.01VLower_{zl} - vp$
pfaref	$pfaref$	Algeb	$-pfaref + pfaref_0$
Qref	$Qref$	Algeb	$-Qref - qref_0$
Qcpf	$Qcpf$	Algeb	$\left(1 - zp_{z1}\right)\left(-Qcpf + S_{1y}\tan\left(pfaref\right)\right)$
PFsel	$PFsel$	Algeb	$-PFsel + QcpfSWPF_{s1} + QrefSWPF_{s0}$
Qerr	$Qerr$	Algeb	$PFlim_{zi}PFsel + PFlim_{zl}QMin + PFlim_{zu}QMax - Qe - Qerr$
PIQ_ys	PIQ_{ys}	Algeb	$\left(1 - Volt_{dip}\right)\left(KqpQerrSWV_{s1} + PIQ_\xi - PIQ_{ys}\right)$
PIQ_y	PIQ_y	Algeb	$\left(1 - Volt_{dip}\right)\left(PIQ_{limzi}PIQ_{ys} + PIQ_{limzl}VMIN + PIQ_{limzu}VMAX - PIQ_y\right)$
Vsel_x	$Vsel_x$	Algeb	$PIQ_ySWV_{s1} + SWV_{s0}\left(QcpfSWPF_{s1} + QrefSWPF_{s0} + Vref_1\right) - Vsel_x$
Verr	$Verr$	Algeb	$-Verr + Vref0r - s_{0y}$
dbV_y	dbV_y	Algeb	$1.0dbV_{dbzl}\left(Verr - dbd_1\right) + 1.0dbV_{dbzu}\left(Verr - dbd_2\right) - dbV_y$
Iqinj	$Iqinj$	Algeb	$-Iqinj + KqvVolt_{dip}dbV_y + fThld\left(1 - Volt_{dip}\right)\left(IqfrzpThld + KqvdbV_ynThld\right)$
wg	wg	Algeb	$1.0 - wg$
Pref	$Pref$	Algeb	$-Pref + \frac{Pref_0}{wg}$
Psel	$Psel$	Algeb	$-Psel + SWP_{s0}pfilt_y + SWP_{s1}pfilt_ywg$
VDL1_y	VDL_{1y}	Algeb	$-VDL_{1y} + \text{FixPiecewise}\left(\left(Iq_1,\ Vq_1 \geq s_{0y}\right), \left(Iq_1 + kVq_{12}\left(-Vq_1 + s_{0y}\right),\ Vq_2 \geq s_0\right.\right.$
VDL2_y	VDL_{2y}	Algeb	$-VDL_{2y} + \text{FixPiecewise}\left(\left(Ip_1,\ Vp_1 \geq s_{0y}\right), \left(Ip_1 + kVp_{12}\left(-Vp_1 + s_{0y}\right),\ Vp_2 \geq s\right.\right.$
Ipmax	$Ipmax$	Algeb	$-Ipmax + IpmaxhfThld_2 + \left(1 - fThld_2\right)\left(\sqrt{Ipmax2sq}SWPQ_{s0} + SWPQ_{s1}\left(z\right.\right.$
Iqmax	$Iqmax$	Algeb	$-Iqmax + \sqrt{Iqmax2sq}SWPQ_{s1} + SWPQ_{s0}\left(zVDL_1\left(Imaxr\left(1 - VDL1c\right) + V\right.\right.$
PIV_ys	PIV_{ys}	Algeb	$\left(1 - Volt_{dip}\right)\left(KvpSWQ_{s1}\left(-SWV_{s0}s_{0y} + VMAXVsel_{limzu} + VMINVsel_{limzl} +\right.\right.$
PIV_y	PIV_y	Algeb	$\left(1 - Volt_{dip}\right)\left(IqmaxPIV_{limzu} + IqminPIV_{limzl} + PIV_{limzi}PIV_{ys} - PIV_y\right)$
Qsel	$Qsel$	Algeb	$PIV_ySWQ_{s1} - Qsel + SWQ_{s0}s_{4y}$
IpHL_x	$IpHL_x$	Algeb	$-IpHL_x + \frac{s_{5y}}{vp}$
IqHL_x	$IqHL_x$	Algeb	$-IqHL_x + Iqinj + Qsel$
a	a	ExtAlgeb	0

Table 13 – continued from p

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
v	v	ExtAlgeb	0
Pe	Pe	ExtAlgeb	0
Qe	Qe	ExtAlgeb	0
Ipcmd	$Ipcmd$	ExtAlgeb	$ue\left(IpHL_{limzi}IpHL_x + IpHL_{limzl}Ipmin + IpHL_{limzu}Ipmax - Ipcmd_0\right)$
Iqcmd	$Iqcmd$	ExtAlgeb	$ue\left(-IqHL_{limzi}IqHL_x - IqHL_{limzl}Iqmin - IqHL_{limzu}Iqmax - Iqcmd_0\right)$

Services

Name	Symbol	Equation
Ipcmd0	$Ipcmd0$	$\frac{p_0}{v}$
Iqcmd0	$Iqcmd0$	$-\frac{q_0}{v}$
pfaref0	Φ_{ref0}	$\text{atan}_2\left(q_0, p_0\right)$
Volt_dip	z_{Vdip}	$1 - Vcmp_{zi}$
qref0	q_{ref0}	$Iqcmd_0 SWQ_{s0}\left(VLower_{zi}v + 0.01VLower_{zl}\right) + SWQ_{s1}\left(-Vref_1 + v\right)$
PIQ_flag	z_{PIQ}^{flag}	0
s4_flag	$z_{s_4}^{flag}$	0
Vref0r	$V_{ref0,r}$	$Vref_0$
pThld	p_{Thld}	Indicator $\left(Thld > 0\right)$
nThld	n_{Thld}	Indicator $\left(Thld < 0\right)$
Thld_abs	$\lvert Thld\rvert$	$\lvert Thld\rvert$
fThld	$fThld$	0
Pref0	P_{ref0}	p_0
s5_flag	z_{s5}^{flag}	0
kVq12	k_{Vq12}	$\frac{-Iq_1+Iq_2}{-Vq_1+Vq_2}$
kVq23	k_{Vq23}	$\frac{-Iq_2+Iq_3}{-Vq_2+Vq_3}$
kVq34	k_{Vq34}	$\frac{-Iq_3+Iq_4}{-Vq_3+Vq_4}$
zVDL1	z_{VDL1}	$Iq_1 \leq Iq_2 \wedge Iq_2 \leq Iq_3 \wedge Iq_3 \leq Iq_4 \wedge Vq_1 \leq Vq_2 \wedge Vq_2 \leq Vq_3 \wedge Vq_3 \leq Vq_4$
kVp12	k_{Vp12}	$\frac{-Ip_1+Ip_2}{-Vp_1+Vp_2}$
kVp23	k_{Vp23}	$\frac{-Ip_2+Ip_3}{-Vp_2+Vp_3}$
kVp34	k_{Vp34}	$\frac{-Ip_3+Ip_4}{-Vp_3+Vp_4}$
zVDL2	z_{VDL2}	$Ip_1 \leq Ip_2 \wedge Ip_2 \leq Ip_3 \wedge Ip_3 \leq Ip_4 \wedge Vp_1 \leq Vp_2 \wedge Vp_2 \leq Vp_3 \wedge Vp_3 \leq Vp_4$
fThld2	$fThld2$	0
VDL1c	$VDL1c$	$VDL_{1y} < Imaxr$
VDL2c	$VDL2c$	$VDL_{2y} < Imaxr$
Ipmax2sq0	$I_{pmax20,nn}^2$	FixPiecewise $\left(\left(0,\ Imax^2 - Iqcmd_0^2 \leq 0\right), \left(Imax^2 - Iqcmd_0^2,\ \text{True}\right)\right)$
Ipmax2sq	I_{pmax2}^2	FixPiecewise $\left(\left(0,\ Imax^2 - \left(IqHL_{limzi}IqHL_x + IqHL_{limzl}Iqmin + IqHL_{limzu}Iqmax\right)^2\right.\right.$
Ipmaxh	$Ipmaxh$	0
Iqmax2sq0	$I_{qmax,nn}^2$	FixPiecewise $\left(\left(0,\ Imax^2 - Ipcmd_0^2 \leq 0\right), \left(Imax^2 - Ipcmd_0^2,\ \text{True}\right)\right)$
Iqmax2sq	I_{qmax2}^2	FixPiecewise $\left(\left(0,\ Imax^2 - \left(IpHL_{limzi}IpHL_x + IpHL_{limzl}Ipmin + IpHL_{limzu}Ipmax\right)^2\right.\right.$

Table 14 – continued fro

Name	Symbol	Equation
Ipmin	I_{pmin}	0.0
PIV_flag	z_{PIV}^{flag}	0

Discretes

Name	Symbol	Type	Info
SWPF	SW_{PF}	Switcher	
SWV	SW_V	Switcher	
SWQ	SW_V	Switcher	
SWP	SW_P	Switcher	
SWPQ	SW_{PQ}	Switcher	
zp	zp	IsEqual	
Vcmp	V_{cmp}	Limiter	Voltage dip comparator
VLower	$VLower$	Limiter	Limiter for lower voltage cap
PFlim	$PFlim$	Limiter	
PIQ_lim	lim_{PIQ}	HardLimiter	
Vsel_lim	lim_{Vsel}	HardLimiter	
dbV_db	db_{dbV}	DeadBand	
pfilt_lim	$lim_{P_{filt}}$	RateLimiter	Rate limiter in Lag
s5_lim	lim_{s5}	AntiWindup	Limiter in Lag
PIV_lim	lim_{PIV}	HardLimiter	
IpHL_lim	lim_{IpHL}	HardLimiter	
IqHL_lim	lim_{IqHL}	HardLimiter	

Blocks

Name	Symbol	Type	Info
s0	$s0$	Lag	Voltage filter
S1	S_1	Lag	Pe filter
PIQ	PIQ	PITrackAWFreeze	
Vsel	$Vsel$	GainLimiter	Selection output of VFLAG
s4	s_4	LagFreeze	Filter for calculated voltage with freeze
dbV	dbV	DeadBand1	Deadband for voltage error (ref0)
pfilt	P_{filt}	LagRate	Active power filter with rate limits
s5	$s5$	LagAWFreeze	
VDL1	V_{DL1}	Piecewise	Piecewise linear characteristics of Vq-Iq
VDL2	V_{DL2}	Piecewise	Piecewise linear characteristics of Vp-Ip
PIV	PIV	PITrackAWFreeze	
IpHL	$IpHL$	GainLimiter	
IqHL	$IqHL$	GainLimiter	

Setpoints

Name	Target	Info
pref	Pref0	initial P ref
qref	qref0	

Config Fields in [REECA1]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)
kqs	K_{qs}	2	Q PI controller tracking gain	
kvs	K_{vs}	2	Voltage PI controller tracking gain	
tpfilt	T_{pfilt}	0.020	Time const. for Pref filter	

REECA1E

REGCA1 with inertia emulation and primary frequency droop. Measurements are based on frequency measurement model.

Bus ROCOF obtained from `BusROCOF` devices.

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
reg		Renewable generator idx			mandatory
busr		Optional remote bus for voltage control			
PFFLAG		Power factor control flag; 1-PF control, 0-Q control		*bool*	mandatory
VFLAG		Voltage control flag; 1-Q control, 0-V control		*bool*	mandatory
QFLAG		Q control flag; 1-V or Q control, 0-const. PF or Q		*bool*	mandatory
PFLAG		P speed-dependency flag; 1-has speed dep., 0-no dep.		*bool*	mandatory
PQFLAG		P/Q priority flag for I limit; 0-Q priority, 1-P priority		*bool*	mandatory
Vdip	V_{dip}	Low V threshold to activate Iqinj logic	0.800	*p.u.*	
Vup	V_{up}	V threshold above which to activate Iqinj logic	1.200	*p.u.*	
Trv	T_{rv}	Voltage filter time constant	0.020		
dbd1	d_{bd1}	Lower bound of the voltage deadband (<=0)	-0.020		
dbd2	d_{bd2}	Upper bound of the voltage deadband (>=0)	0.020		
Kqv	K_{qv}	Gain to compute Iqinj from V error (caution!!)	1		
Iqh1	I_{qh1}	Upper limit on Iqinj	999		
Iql1	I_{ql1}	Lower limit on Iqinj	-999		
Vref0	V_{ref0}	User defined Vref (if 0, use initial bus V)	1		
Iqfrz	I_{qfrz}	Hold Iqinj at the value for Thld (>0) seconds following a Vdip	0		
Thld	T_{hld}	Time for which Iqinj is held. Hold at Iqinj if>0; hold at State 1 if<0	0	*s*	
Thld2	T_{hld2}	Time for which IPMAX is held after voltage dip ends	0	*s*	
Tp	T_p	Filter time constant for Pe	0.020	*s*	
QMax	Q_{max}	Upper limit for reactive power regulator	999		power
QMin	Q_{min}	Lower limit for reactive power regulator	-999		power
VMAX	V_{max}	Upper limit for voltage control	999		
VMIN	V_{min}	Lower limit for voltage control	-999		
Kqp	K_{qp}	Proportional gain for reactive power error	1		
Kqi	K_{qi}	Integral gain for reactive power error	0.100		
Kvp	K_{vp}	Proportional gain for voltage error	1		
Kvi	K_{vi}	Integral gain for voltage error	0.100		
Vref1	V_{ref1}	Voltage ref. if VFLAG=0	1		non_zero
Tiq	T_{iq}	Filter time constant for Iq	0.020		
dPmax	d_{Pmax}	Power reference max. ramp rate (>0)	999		power
dPmin	d_{Pin}	Power reference min. ramp rate (<0)	-999		power

continues on next page

Table 15 – continued from previous page

Name	Symbol	Description	Default	Unit	Properties
PMAX	P_{max}	Max. active power limit > 0	999		power
PMIN	P_{min}	Min. active power limit	0		power
Imax	I_{max}	Max. apparent current limit	999		current
Tpord	T_{pord}	Filter time constant for power setpoint	0.020		
Vq1	V_{q1}	Reactive power V-I pair (point 1), voltage	0.200		
Iq1	I_{q1}	Reactive power V-I pair (point 1), current	2		current
Vq2	V_{q2}	Reactive power V-I pair (point 2), voltage	0.400		
Iq2	I_{q2}	Reactive power V-I pair (point 2), current	4		current
Vq3	V_{q3}	Reactive power V-I pair (point 3), voltage	0.800		
Iq3	I_{q3}	Reactive power V-I pair (point 3), current	8		current
Vq4	V_{q4}	Reactive power V-I pair (point 4), voltage	1		
Iq4	I_{q4}	Reactive power V-I pair (point 4), current	10		current
Vp1	V_{p1}	Active power V-I pair (point 1), voltage	0.200		
Ip1	I_{p1}	Active power V-I pair (point 1), current	2		current
Vp2	V_{p2}	Active power V-I pair (point 2), voltage	0.400		
Ip2	I_{p2}	Active power V-I pair (point 2), current	4		current
Vp3	V_{p3}	Active power V-I pair (point 3), voltage	0.800		
Ip3	I_{p3}	Active power V-I pair (point 3), current	8		current
Vp4	V_{p4}	Active power V-I pair (point 4), voltage	1		
Ip4	I_{p4}	Active power V-I pair (point 4), current	12		current
Kf	K_{df}	gain for frequency deviation	0		
Kdf	K_{df}	gain for rate-of-change of frequency	0		
busroc		Optional BusROCOF device idx			
ue	u_e	effective online status	0		
bus		Retrieved bus idx			
gen		Retrieved StaticGen idx			
Sn	S_n		0		

Variables

Name	Symbol	Type	Description	Unit	Properties
s0_y	s_{0y}	State	State in lag transfer function		v_str
S1_y	S_{1y}	State	State in lag transfer function		v_str
PIQ_xi	PIQ_{ξ}	State	Integrator output		v_str
s4_y	s_{4y}	State	State in lag transfer function		v_str
pfilt_y	$pfilt_y$	State	State in lag TF		v_str
s5_y	s_{5y}	State	State in lag TF		v_str
PIV_xi	PIV_{ξ}	State	Integrator output		v_str
Pord	$Pord$	AliasState	Alias of s5_y		
vp	vp	Algeb	Sensed lower-capped voltage		v_str
pfaref	$pfaref$	Algeb	power factor angle ref	*rad*	v_str

continues on next page

Table 16 – continued from previous page

Name	Symbol	Type	Description	Unit	Properties
Qref	$Qref$	Algeb	external Q ref	*p.u.*	v_str
Qcpf	$Qcpf$	Algeb	Q calculated from P and power factor	*p.u.*	v_str
PFsel	$PFsel$	Algeb	Output of PFFLAG selector		v_str
Qerr	$Qerr$	Algeb	Reactive power error		v_str
PIQ_ys	PIQ_{ys}	Algeb	PI summation before limit		v_str
PIQ_y	PIQ_y	Algeb	PI output		v_str
Vsel_x	$Vsel_x$	Algeb	Value before limiter		v_str
Verr	$Verr$	Algeb	Voltage error (Vref0)		v_str
dbV_y	dbV_y	Algeb	Deadband type 1 output		v_str
Iqinj	$Iqinj$	Algeb	Additional Iq signal during under- or over-voltage		v_str
wg	wg	Algeb	Drive train generator speed		v_str
Pref	$Pref$	Algeb	external P ref	*p.u.*	v_str
Psel	$Psel$	Algeb	Output selection of PFLAG		v_str
VDL1_y	VDL_{1y}	Algeb	Output of piecewise		v_str
VDL2_y	VDL_{2y}	Algeb	Output of piecewise		v_str
Ipmax	$Ipmax$	Algeb	Upper limit on Ipcmd		v_str
Iqmax	$Iqmax$	Algeb	Upper limit on Iqcmd		v_str
PIV_ys	PIV_{ys}	Algeb	PI summation before limit		v_str
PIV_y	PIV_y	Algeb	PI output		v_str
Qsel	$Qsel$	Algeb	Selection output of QFLAG		v_str
IpHL_x	$IpHL_x$	Algeb	Value before limiter		v_str
IqHL_x	$IqHL_x$	Algeb	Value before limiter		v_str
a	a	ExtAlgeb	Bus voltage angle		
v	v	ExtAlgeb	d-axis bus voltage magnitude		
Pe	Pe	ExtAlgeb	Retrieved Pe of RenGen		
Qe	Qe	ExtAlgeb	Retrieved Qe of RenGen		
Ipcmd	$Ipcmd$	ExtAlgeb	Retrieved Ipcmd of RenGen		
Iqcmd	$Iqcmd$	ExtAlgeb	Retrieved Iqcmd of RenGen		
df	df	ExtAlgeb	Bus frequency deviation		
dfdt	$dfdt$	ExtAlgeb	Bus ROCOF	*p.u.*	

Initialization Equations

Name	Symbol	Type	Initial Value
s0_y	s_{0y}	State	v
S1_y	S_{1y}	State	Pe
PIQ_xi	PIQ_ξ	State	0.0
s4_y	s_{4y}	State	$\frac{PFsel}{vp}$
pfilt_y	$pfilt_y$	State	$Pref$
s5_y	s_{5y}	State	$Psel$
PIV_xi	PIV_ξ	State	$-Iqcmd_0 SWQ_{s1}$

Table 17 – continued from previo

Name	Symbol	Type	Initial Value
Pord	$Pord$	AliasState	
vp	vp	Algeb	$VLower_{zi}v + 0.01VLower_{zl}$
pfaref	$pfaref$	Algeb	$pfaref_0$
Qref	$Qref$	Algeb	$-qref_0$
Qcpf	$Qcpf$	Algeb	q_0
PFsel	$PFsel$	Algeb	$QcpfSWPF_{s1} + QrefSWPF_{s0}$
Qerr	$Qerr$	Algeb	$PFlim_{zi}PFsel + PFlim_{zl}QMin + PFlim_{zu}QMax - Qe$
PIQ_ys	PIQ_{ys}	Algeb	$KqpQerrSWV_{s1}$
PIQ_y	PIQ_y	Algeb	$PIQ_{limzi}PIQ_{ys} + PIQ_{limzl}VMIN + PIQ_{limzu}VMAX$
Vsel_x	$Vsel_x$	Algeb	$PIQ_ySWV_{s1} + SWV_{s0}\left(QcpfSWPF_{s1} + QrefSWPF_{s0} + Vref_1\right)$
Verr	$Verr$	Algeb	$Vref0r - s_{0y}$
dbV_y	dbV_y	Algeb	$1.0dbV_{dbzl}\left(Verr - dbd_1\right) + 1.0dbV_{dbzu}\left(Verr - dbd_2\right)$
Iqinj	$Iqinj$	Algeb	$KqvVolt_{dip}dbV_y + fThld\left(1 - Volt_{dip}\right)\left(IqfrzpThld + KqvdbV_ynThld\right)$
wg	wg	Algeb	1.0
Pref	$Pref$	Algeb	$\frac{p_0}{wg}$
Psel	$Psel$	Algeb	$SWP_{s0}pfilt_y + SWP_{s1}pfilt_ywg$
VDL1_y	VDL_{1y}	Algeb	FixPiecewise $\left(\left(Iq_1,\ Vq_1 \geq s_{0y}\right), \left(Iq_1 + kVq_{12}\left(-Vq_1 + s_{0y}\right),\ Vq_2 \geq s_{0y}\right), \left(Iq_2 + k\right.\right.$
VDL2_y	VDL_{2y}	Algeb	FixPiecewise $\left(\left(Ip_1,\ Vp_1 \geq s_{0y}\right), \left(Ip_1 + kVp_{12}\left(-Vp_1 + s_{0y}\right),\ Vp_2 \geq s_{0y}\right), \left(Ip_2 + \right.\right.$
Ipmax	$Ipmax$	Algeb	$\left(1 - fThld_2\right)\left(\sqrt{Ipmax2sq0}SWPQ_{s0} + SWPQ_{s1}\left(zVDL_2\left(Imaxr\left(1 - VDL2c\right)\right.\right.\right.$
Iqmax	$Iqmax$	Algeb	$\sqrt{Iqmax2sq0}SWPQ_{s1} + SWPQ_{s0}\left(zVDL_1\left(Imaxr\left(1 - VDL1c\right) + VDL_{1y}VDL\right.\right.$
PIV_ys	PIV_{ys}	Algeb	$-Iqcmd_0SWQ_{s1} + KvpSWQ_{s1}\left(-SWV_{s0}s_{0y} + VMAXVsel_{limzu} + VMINVse\right.$
PIV_y	PIV_y	Algeb	$IqmaxPIV_{limzu} + IqminPIV_{limzl} + PIV_{limzi}PIV_{ys}$
Qsel	$Qsel$	Algeb	$PIV_ySWQ_{s1} + SWQ_{s0}s_{4y}$
IpHL_x	$IpHL_x$	Algeb	$\frac{s_{5y}}{vp}$
IqHL_x	$IqHL_x$	Algeb	$Iqinj + Qsel$
a	a	ExtAlgeb	
v	v	ExtAlgeb	
Pe	Pe	ExtAlgeb	
Qe	Qe	ExtAlgeb	
Ipcmd	$Ipcmd$	ExtAlgeb	
Iqcmd	$Iqcmd$	ExtAlgeb	
df	df	ExtAlgeb	
dfdt	$dfdt$	ExtAlgeb	

Differential Equations

Nam	Sym-bol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
s0_y	s_{0y}	State	$-s_{0y} + v$	T_{rv}
S1_y	S_{1y}	State	$Pe - S_{1y}$	T_p
PIQ_	PIQ_ξ	State	$Kqi\left(1 - Volt_{dip}\right)\left(2PIQ_y - 2PIQ_{ys} + QerrSWV_{s1}\right)$	
s4_y	s_{4y}	State	$\left(1 - Volt_{dip}\right)\left(\frac{PFsel}{vp} - s_{4y}\right)$	T_{iq}
pfilt_	$pfilt_y$	State	$Pref - pfilt_y$	0.02
s5_y	s_{5y}	State	$\left(1 - Volt_{dip}\right)\left(Psel - s_{5y}\right)$	T_{pord}
PIV_	PIV_ξ	State	$Kvi\left(1 - Volt_{dip}\right)\left(2PIV_y - 2PIV_{ys} + SWQ_{s1}\left(-SWV_{s0}s_{0y} + VMAX\right.\right.$	$_{nzu} + VMINV$
Pord	$Pord$	AliasSt	0	

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
vp	vp	Algeb	$VLower_{zi}v + 0.01VLower_{zl} - vp$
pfaref	$pfaref$	Algeb	$-pfaref + pfaref_0$
Qref	$Qref$	Algeb	$-Qref - qref_0$
Qcpf	$Qcpf$	Algeb	$\left(1 - zp_{z1}\right)\left(-Qcpf + S_{1y}\tan\left(pfaref\right)\right)$
PFsel	$PFsel$	Algeb	$-PFsel + QcpfSWPF_{s1} + QrefSWPF_{s0}$
Qerr	$Qerr$	Algeb	$PFlim_{zi}PFsel + PFlim_{zl}QMin + PFlim_{zu}QMax - Qe - Qerr$
PIQ_ys	PIQ_{ys}	Algeb	$\left(1 - Volt_{dip}\right)\left(KqpQerrSWV_{s1} + PIQ_\xi - PIQ_{ys}\right)$
PIQ_y	PIQ_y	Algeb	$\left(1 - Volt_{dip}\right)\left(PIQ_{limzi}PIQ_{ys} + PIQ_{limzl}VMIN + PIQ_{limzu}VMAX - PIQ_y\right)$
Vsel_x	$Vsel_x$	Algeb	$PIQ_ySWV_{s1} + SWV_{s0}\left(QcpfSWPF_{s1} + QrefSWPF_{s0} + Vref_1\right) - Vsel_x$
Verr	$Verr$	Algeb	$-Verr + Vref0r - s_{0y}$
dbV_y	dbV_y	Algeb	$1.0dbV_{dbzl}\left(Verr - dbd_1\right) + 1.0dbV_{dbzu}\left(Verr - dbd_2\right) - dbV_y$
Iqinj	$Iqinj$	Algeb	$-Iqinj + KqvVolt_{dip}dbV_y + fThld\left(1 - Volt_{dip}\right)\left(IqfrzpThld + KqvdbV_ynThld\right)$
wg	wg	Algeb	$1.0 - wg$
Pref	$Pref$	Algeb	$-Kdfdfdt - Kfdf - Pref + \frac{Pref_0}{wg}$
Psel	$Psel$	Algeb	$-Psel + SWP_{s0}pfilt_y + SWP_{s1}pfilt_ywg$
VDL1_y	VDL_{1y}	Algeb	$-VDL_{1y} + \text{FixPiecewise}\left(\left(Iq_1,\ Vq_1 \geq s_{0y}\right), \left(Iq_1 + kVq_{12}\left(-Vq_1 + s_{0y}\right),\ Vq_2 \geq s_0\right.\right.$
VDL2_y	VDL_{2y}	Algeb	$-VDL_{2y} + \text{FixPiecewise}\left(\left(Ip_1,\ Vp_1 \geq s_{0y}\right), \left(Ip_1 + kVp_{12}\left(-Vp_1 + s_{0y}\right),\ Vp_2 \geq s\right.\right.$
Ipmax	$Ipmax$	Algeb	$-Ipmax + IpmaxhfThld_2 + \left(1 - fThld_2\right)\left(\sqrt{Ipmax2sq}SWPQ_{s0} + SWPQ_{s1}\left(z\right.\right.$
Iqmax	$Iqmax$	Algeb	$-Iqmax + \sqrt{Iqmax2sq}SWPQ_{s1} + SWPQ_{s0}\left(zVDL_1\left(Imaxr\left(1 - VDL1c\right) + V\right.\right.$
PIV_ys	PIV_{ys}	Algeb	$\left(1 - Volt_{dip}\right)\left(KvpSWQ_{s1}\left(-SWV_{s0}s_{0y} + VMAXVsel_{limzu} + VMINVsel_{limzl} +\right.\right.$
PIV_y	PIV_y	Algeb	$\left(1 - Volt_{dip}\right)\left(IqmaxPIV_{limzu} + IqminPIV_{limzl} + PIV_{limzi}PIV_{ys} - PIV_y\right)$
Qsel	$Qsel$	Algeb	$PIV_ySWQ_{s1} - Qsel + SWQ_{s0}s_{4y}$
IpHL_x	$IpHL_x$	Algeb	$-IpHL_x + \frac{s_{5y}}{vp}$
IqHL_x	$IqHL_x$	Algeb	$-IqHL_x + Iqinj + Qsel$
a	a	ExtAlgeb	0

Table 18 – continued from p

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
v	v	ExtAlgeb	0
Pe	Pe	ExtAlgeb	0
Qe	Qe	ExtAlgeb	0
Ipcmd	$Ipcmd$	ExtAlgeb	$ue\left(IpHL_{limzi}IpHL_x + IpHL_{limzl}Ipmin + IpHL_{limzu}Ipmax - Ipcmd_0\right)$
Iqcmd	$Iqcmd$	ExtAlgeb	$ue\left(-IqHL_{limzi}IqHL_x - IqHL_{limzl}Iqmin - IqHL_{limzu}Iqmax - Iqcmd_0\right)$
df	df	ExtAlgeb	0
dfdt	$dfdt$	ExtAlgeb	0

Services

Name	Symbol	Equation
Ipcmd0	$Ipcmd0$	$\frac{p_0}{v}$
Iqcmd0	$Iqcmd0$	$-\frac{q_0}{v}$
pfaref0	Φ_{ref0}	$\text{atan}_2\left(q_0, p_0\right)$
Volt_dip	z_{Vdip}	$1 - Vcmp_{zi}$
qref0	q_{ref0}	$Iqcmd_0 SWQ_{s0}\left(VLower_{zi}v + 0.01VLower_{zl}\right) + SWQ_{s1}\left(-Vref_1 + v\right)$
PIQ_flag	z_{PIQ}^{flag}	0
s4_flag	z_{s4}^{flag}	0
Vref0r	$V_{ref0,r}$	$Vref_0$
pThld	p_{Thld}	Indicator $(Thld > 0)$
nThld	n_{Thld}	Indicator $(Thld < 0)$
Thld_abs	$\lvert Thld\rvert$	$\lvert Thld\rvert$
fThld	$fThld$	0
Pref0	P_{ref0}	p_0
s5_flag	z_{s5}^{flag}	0
kVq12	k_{Vq12}	$\frac{-Iq_1+Iq_2}{-Vq_1+Vq_2}$
kVq23	k_{Vq23}	$\frac{-Iq_2+Iq_3}{-Vq_2+Vq_3}$
kVq34	k_{Vq34}	$\frac{-Iq_3+Iq_4}{-Vq_3+Vq_4}$
zVDL1	z_{VDL1}	$Iq_1 \leq Iq_2 \wedge Iq_2 \leq Iq_3 \wedge Iq_3 \leq Iq_4 \wedge Vq_1 \leq Vq_2 \wedge Vq_2 \leq Vq_3 \wedge Vq_3 \leq Vq_4$
kVp12	k_{Vp12}	$\frac{-Ip_1+Ip_2}{-Vp_1+Vp_2}$
kVp23	k_{Vp23}	$\frac{-Ip_2+Ip_3}{-Vp_2+Vp_3}$
kVp34	k_{Vp34}	$\frac{-Ip_3+Ip_4}{-Vp_3+Vp_4}$
zVDL2	z_{VDL2}	$Ip_1 \leq Ip_2 \wedge Ip_2 \leq Ip_3 \wedge Ip_3 \leq Ip_4 \wedge Vp_1 \leq Vp_2 \wedge Vp_2 \leq Vp_3 \wedge Vp_3 \leq Vp_4$
fThld2	$fThld2$	0
VDL1c	$VDL1c$	$VDL_{1y} < Imaxr$
VDL2c	$VDL2c$	$VDL_{2y} < Imaxr$
Ipmax2sq0	$I_{pmax20,nn}^2$	FixPiecewise $\left(\left(0,\ Imax^2 - Iqcmd_0^2 \leq 0\right), \left(Imax^2 - Iqcmd_0^2,\ \text{True}\right)\right)$
Ipmax2sq	I_{pmax2}^2	FixPiecewise $\Big(\Big(0,\ Imax^2 - \left(IqHL_{limzi}IqHL_x + IqHL_{limzl}Iqmin + IqHL_{limzu}Iqmax\right)^2$
Ipmaxh	$Ipmaxh$	0

Table 19 – continued fro

Name	Symbol	Equation
Iqmax2sq0	$I_{qmax,nn}^2$	FixPiecewise $\left(\left(0,\ Imax^2 - Ipcmd_0^2 \leq 0\right), \left(Imax^2 - Ipcmd_0^2,\ \text{True}\right)\right)$
Iqmax2sq	I_{qmax2}^2	FixPiecewise $\left(\left(0,\ Imax^2 - (IpHL_{limzi}IpHL_x + IpHL_{limzl}Ipmin + IpHL_{limzu}Ipmax)^2\right.\right.$
Ipmin	I_{pmin}	0.0
PIV_flag	z_{PIV}^{flag}	0

Discretes

Name	Symbol	Type	Info
SWPF	SW_{PF}	Switcher	
SWV	SW_V	Switcher	
SWQ	SW_V	Switcher	
SWP	SW_P	Switcher	
SWPQ	SW_{PQ}	Switcher	
zp	zp	IsEqual	
Vcmp	V_{cmp}	Limiter	Voltage dip comparator
VLower	$VLower$	Limiter	Limiter for lower voltage cap
PFlim	$PFlim$	Limiter	
PIQ_lim	lim_{PIQ}	HardLimiter	
Vsel_lim	lim_{Vsel}	HardLimiter	
dbV_db	db_{dbV}	DeadBand	
pfilt_lim	$lim_{P_{filt}}$	RateLimiter	Rate limiter in Lag
s5_lim	lim_{s5}	AntiWindup	Limiter in Lag
PIV_lim	lim_{PIV}	HardLimiter	
IpHL_lim	lim_{IpHL}	HardLimiter	
IqHL_lim	lim_{IqHL}	HardLimiter	

Blocks

Name	Symbol	Type	Info
s0	$s0$	Lag	Voltage filter
S1	S_1	Lag	Pe filter
PIQ	PIQ	PITrackAWFreeze	
Vsel	$Vsel$	GainLimiter	Selection output of VFLAG
s4	s_4	LagFreeze	Filter for calculated voltage with freeze
dbV	dbV	DeadBand1	Deadband for voltage error (ref0)
pfilt	P_{filt}	LagRate	Active power filter with rate limits
s5	$s5$	LagAWFreeze	
VDL1	V_{DL1}	Piecewise	Piecewise linear characteristics of Vq-Iq
VDL2	V_{DL2}	Piecewise	Piecewise linear characteristics of Vp-Ip
PIV	PIV	PITrackAWFreeze	
IpHL	$IpHL$	GainLimiter	
IqHL	$IqHL$	GainLimiter	

Setpoints

Name	Target	Info
pref	Pref0	initial P ref
qref	qref0	

Config Fields in [REECA1E]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)
kqs	K_{qs}	2	Q PI controller tracking gain	
kvs	K_{vs}	2	Voltage PI controller tracking gain	
tpfilt	T_{pfilt}	0.020	Time const. for Pref filter	

REECA1G

REECA1G is a variant of REECA1E.

REECA1G uses speed from synchronous generators.

The application of this model is limited because it is uncommon to connect a SynGen on the same bus as a RenGen.

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
reg		Renewable generator idx			mandatory
busr		Optional remote bus for voltage control			
PFFLAG		Power factor control flag; 1-PF control, 0-Q control		*bool*	mandatory
VFLAG		Voltage control flag; 1-Q control, 0-V control		*bool*	mandatory
QFLAG		Q control flag; 1-V or Q control, 0-const. PF or Q		*bool*	mandatory
PFLAG		P speed-dependency flag; 1-has speed dep., 0-no dep.		*bool*	mandatory
PQFLAG		P/Q priority flag for I limit; 0-Q priority, 1-P priority		*bool*	mandatory
Vdip	V_{dip}	Low V threshold to activate Iqinj logic	0.800	*p.u.*	
Vup	V_{up}	V threshold above which to activate Iqinj logic	1.200	*p.u.*	
Trv	T_{rv}	Voltage filter time constant	0.020		
dbd1	d_{bd1}	Lower bound of the voltage deadband (<=0)	-0.020		
dbd2	d_{bd2}	Upper bound of the voltage deadband (>=0)	0.020		
Kqv	K_{qv}	Gain to compute Iqinj from V error (caution!!)	1		
Iqh1	I_{qh1}	Upper limit on Iqinj	999		
Iql1	I_{ql1}	Lower limit on Iqinj	-999		
Vref0	V_{ref0}	User defined Vref (if 0, use initial bus V)	1		
Iqfrz	I_{qfrz}	Hold Iqinj at the value for Thld (>0) seconds following a Vdip	0		
Thld	T_{hld}	Time for which Iqinj is held. Hold at Iqinj if>0; hold at State 1 if<0	0	*s*	
Thld2	T_{hld2}	Time for which IPMAX is held after voltage dip ends	0	*s*	
Tp	T_p	Filter time constant for Pe	0.020	*s*	
QMax	Q_{max}	Upper limit for reactive power regulator	999		power
QMin	Q_{min}	Lower limit for reactive power regulator	-999		power
VMAX	V_{max}	Upper limit for voltage control	999		
VMIN	V_{min}	Lower limit for voltage control	-999		
Kqp	K_{qp}	Proportional gain for reactive power error	1		
Kqi	K_{qi}	Integral gain for reactive power error	0.100		
Kvp	K_{vp}	Proportional gain for voltage error	1		
Kvi	K_{vi}	Integral gain for voltage error	0.100		
Vref1	V_{ref1}	Voltage ref. if VFLAG=0	1		non_zero
Tiq	T_{iq}	Filter time constant for Iq	0.020		

continues on next page

Table 20 – continued from previous page

Name	Symbol	Description	Default	Unit	Properties
dPmax	d_{Pmax}	Power reference max. ramp rate (>0)	999		power
dPmin	d_{Pin}	Power reference min. ramp rate (<0)	-999		power
PMAX	P_{max}	Max. active power limit > 0	999		power
PMIN	P_{min}	Min. active power limit	0		power
Imax	I_{max}	Max. apparent current limit	999		current
Tpord	T_{pord}	Filter time constant for power setpoint	0.020		
Vq1	V_{q1}	Reactive power V-I pair (point 1), voltage	0.200		
Iq1	I_{q1}	Reactive power V-I pair (point 1), current	2		current
Vq2	V_{q2}	Reactive power V-I pair (point 2), voltage	0.400		
Iq2	I_{q2}	Reactive power V-I pair (point 2), current	4		current
Vq3	V_{q3}	Reactive power V-I pair (point 3), voltage	0.800		
Iq3	I_{q3}	Reactive power V-I pair (point 3), current	8		current
Vq4	V_{q4}	Reactive power V-I pair (point 4), voltage	1		
Iq4	I_{q4}	Reactive power V-I pair (point 4), current	10		current
Vp1	V_{p1}	Active power V-I pair (point 1), voltage	0.200		
Ip1	I_{p1}	Active power V-I pair (point 1), current	2		current
Vp2	V_{p2}	Active power V-I pair (point 2), voltage	0.400		
Ip2	I_{p2}	Active power V-I pair (point 2), current	4		current
Vp3	V_{p3}	Active power V-I pair (point 3), voltage	0.800		
Ip3	I_{p3}	Active power V-I pair (point 3), current	8		current
Vp4	V_{p4}	Active power V-I pair (point 4), voltage	1		
Ip4	I_{p4}	Active power V-I pair (point 4), current	12		current
Kf	K_{df}	gain for frequency deviation	0		
sg		synchronous gen idx			mandatory
ue	u_e	effective online status	0		
bus		Retrieved bus idx			
gen		Retrieved StaticGen idx			
Sn	S_n		0		

Variables

Name	Symbol	Type	Description	Unit	Properties
s0_y	s_{0y}	State	State in lag transfer function		v_str
S1_y	S_{1y}	State	State in lag transfer function		v_str
PIQ_xi	PIQ_{ξ}	State	Integrator output		v_str
s4_y	s_{4y}	State	State in lag transfer function		v_str
pfilt_y	$pfilt_y$	State	State in lag TF		v_str
s5_y	s_{5y}	State	State in lag TF		v_str
PIV_xi	PIV_{ξ}	State	Integrator output		v_str
Pord	$Pord$	AliasState	Alias of s5_y		
omega	ω	ExtState	generator speed	*pu*	

continues on next page

Table 21 – continued from previous page

Name	Symbol	Type	Description	Unit	Properties
vp	vp	Algeb	Sensed lower-capped voltage		v_str
pfaref	$pfaref$	Algeb	power factor angle ref	*rad*	v_str
Qref	$Qref$	Algeb	external Q ref	*p.u.*	v_str
Qcpf	$Qcpf$	Algeb	Q calculated from P and power factor	*p.u.*	v_str
PFsel	$PFsel$	Algeb	Output of PFFLAG selector		v_str
Qerr	$Qerr$	Algeb	Reactive power error		v_str
PIQ_ys	PIQ_{ys}	Algeb	PI summation before limit		v_str
PIQ_y	PIQ_y	Algeb	PI output		v_str
Vsel_x	$Vsel_x$	Algeb	Value before limiter		v_str
Verr	$Verr$	Algeb	Voltage error (Vref0)		v_str
dbV_y	dbV_y	Algeb	Deadband type 1 output		v_str
Iqinj	$Iqinj$	Algeb	Additional Iq signal during under- or over-voltage		v_str
wg	wg	Algeb	Drive train generator speed		v_str
Pref	$Pref$	Algeb	external P ref	*p.u.*	v_str
Psel	$Psel$	Algeb	Output selection of PFLAG		v_str
VDL1_y	VDL_{1y}	Algeb	Output of piecewise		v_str
VDL2_y	VDL_{2y}	Algeb	Output of piecewise		v_str
Ipmax	$Ipmax$	Algeb	Upper limit on Ipcmd		v_str
Iqmax	$Iqmax$	Algeb	Upper limit on Iqcmd		v_str
PIV_ys	PIV_{ys}	Algeb	PI summation before limit		v_str
PIV_y	PIV_y	Algeb	PI output		v_str
Qsel	$Qsel$	Algeb	Selection output of QFLAG		v_str
IpHL_x	$IpHL_x$	Algeb	Value before limiter		v_str
IqHL_x	$IqHL_x$	Algeb	Value before limiter		v_str
a	a	ExtAlgeb	Bus voltage angle		
v	v	ExtAlgeb	d-axis bus voltage magnitude		
Pe	Pe	ExtAlgeb	Retrieved Pe of RenGen		
Qe	Qe	ExtAlgeb	Retrieved Qe of RenGen		
Ipcmd	$Ipcmd$	ExtAlgeb	Retrieved Ipcmd of RenGen		
Iqcmd	$Iqcmd$	ExtAlgeb	Retrieved Iqcmd of RenGen		

Initialization Equations

Name	Symbol	Type	Initial Value
s0_y	s_{0y}	State	v
S1_y	S_{1y}	State	Pe
PIQ_xi	PIQ_ξ	State	0.0
s4_y	s_{4y}	State	$\frac{PFsel}{vp}$
pfilt_y	$pfilt_y$	State	$Pref$
s5_y	s_{5y}	State	$Psel$
PIV_xi	PIV_ξ	State	$-Iqcmd_0 SWQ_{s1}$

Table 22 – continued from previo

Name	Symbol	Type	Initial Value
Pord	$Pord$	AliasState	
omega	ω	ExtState	
vp	vp	Algeb	$VLower_{zi}v + 0.01VLower_{zl}$
pfaref	$pfaref$	Algeb	$pfaref_0$
Qref	$Qref$	Algeb	$-qref_0$
Qcpf	$Qcpf$	Algeb	q_0
PFsel	$PFsel$	Algeb	$QcpfSWPF_{s1} + QrefSWPF_{s0}$
Qerr	$Qerr$	Algeb	$PFlim_{zi}PFsel + PFlim_{zl}QMin + PFlim_{zu}QMax - Qe$
PIQ_ys	PIQ_{ys}	Algeb	$KqpQerrSWV_{s1}$
PIQ_y	PIQ_y	Algeb	$PIQ_{limzi}PIQ_{ys} + PIQ_{limzl}VMIN + PIQ_{limzu}VMAX$
Vsel_x	$Vsel_x$	Algeb	$PIQ_ySWV_{s1} + SWV_{s0}\left(QcpfSWPF_{s1} + QrefSWPF_{s0} + Vref_1\right)$
Verr	$Verr$	Algeb	$Vref0r - s_{0y}$
dbV_y	dbV_y	Algeb	$1.0dbV_{dbzl}\left(Verr - dbd_1\right) + 1.0dbV_{dbzu}\left(Verr - dbd_2\right)$
Iqinj	$Iqinj$	Algeb	$KqvVolt_{dip}dbV_y + fThld\left(1 - Volt_{dip}\right)\left(IqfrzpThld + KqvdbV_ynThld\right)$
wg	wg	Algeb	1.0
Pref	$Pref$	Algeb	$\frac{p_0}{wg}$
Psel	$Psel$	Algeb	$SWP_{s0}pfilt_y + SWP_{s1}pfilt_ywg$
VDL1_y	VDL_{1y}	Algeb	FixPiecewise $\left(\left(Iq_1,\ Vq_1 \geq s_{0y}\right), \left(Iq_1 + kVq_{12}\left(-Vq_1 + s_{0y}\right),\ Vq_2 \geq s_{0y}\right), \left(Iq_2 + k\right.\right.$
VDL2_y	VDL_{2y}	Algeb	FixPiecewise $\left(\left(Ip_1,\ Vp_1 \geq s_{0y}\right), \left(Ip_1 + kVp_{12}\left(-Vp_1 + s_{0y}\right),\ Vp_2 \geq s_{0y}\right), \left(Ip_2 + \right.\right.$
Ipmax	$Ipmax$	Algeb	$\left(1 - fThld_2\right)\left(\sqrt{Ipmax2sq0}SWPQ_{s0} + SWPQ_{s1}\left(zVDL_2\left(Imaxr\left(1 - VDL2c\right)\right.\right.\right.$
Iqmax	$Iqmax$	Algeb	$\sqrt{Iqmax2sq0}SWPQ_{s1} + SWPQ_{s0}\left(zVDL_1\left(Imaxr\left(1 - VDL1c\right) + VDL_{1y}VDL\right.\right.$
PIV_ys	PIV_{ys}	Algeb	$-Iqcmd_0SWQ_{s1} + KvpSWQ_{s1}\left(-SWV_{s0}s_{0y} + VMAXVsel_{limzu} + VMINVsel\right.$
PIV_y	PIV_y	Algeb	$IqmaxPIV_{limzu} + IqminPIV_{limzl} + PIV_{limzi}PIV_{ys}$
Qsel	$Qsel$	Algeb	$PIV_ySWQ_{s1} + SWQ_{s0}s_{4y}$
IpHL_x	$IpHL_x$	Algeb	$\frac{s_{5y}}{vp}$
IqHL_x	$IqHL_x$	Algeb	$Iqinj + Qsel$
a	a	ExtAlgeb	
v	v	ExtAlgeb	
Pe	Pe	ExtAlgeb	
Qe	Qe	ExtAlgeb	
Ipcmd	$Ipcmd$	ExtAlgeb	
Iqcmd	$Iqcmd$	ExtAlgeb	

Differential Equations

Nam	Sym-bol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
s0_y	s_{0y}	State	$-s_{0y} + v$	T_{rv}
S1_y	S_{1y}	State	$Pe - S_{1y}$	T_p
PIQ_	PIQ_ξ	State	$Kqi\left(1 - Volt_{dip}\right)\left(2PIQ_y - 2PIQ_{ys} + QerrSWV_{s1}\right)$	
s4_y	s_{4y}	State	$\left(1 - Volt_{dip}\right)\left(\frac{PFsel}{vp} - s_{4y}\right)$	T_{iq}
pfilt_	$pfilt_y$	State	$Pref - pfilt_y$	0.02
s5_y	s_{5y}	State	$\left(1 - Volt_{dip}\right)\left(Psel - s_{5y}\right)$	T_{pord}
PIV_	PIV_ξ	State	$Kvi\left(1 - Volt_{dip}\right)\left(2PIV_y - 2PIV_{ys} + SWQ_{s1}\left(-SWV_{s0}s_{0y} + VMAX\right.\right.$	$_{nzu} + VMINV$
Pord	$Pord$	AliasSt	0	
omeg	ω	ExtStat	0	

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
vp	vp	Algeb	$VLower_{zi}v + 0.01VLower_{zl} - vp$
pfaref	$pfaref$	Algeb	$-pfaref + pfaref_0$
Qref	$Qref$	Algeb	$-Qref - qref_0$
Qcpf	$Qcpf$	Algeb	$\left(1 - zp_{z1}\right)\left(-Qcpf + S_{1y}\tan\left(pfaref\right)\right)$
PFsel	$PFsel$	Algeb	$-PFsel + QcpfSWPF_{s1} + QrefSWPF_{s0}$
Qerr	$Qerr$	Algeb	$PFlim_{zi}PFsel + PFlim_{zl}QMin + PFlim_{zu}QMax - Qe - Qerr$
PIQ_ys	PIQ_{ys}	Algeb	$\left(1 - Volt_{dip}\right)\left(KqpQerrSWV_{s1} + PIQ_\xi - PIQ_{ys}\right)$
PIQ_y	PIQ_y	Algeb	$\left(1 - Volt_{dip}\right)\left(PIQ_{limzi}PIQ_{ys} + PIQ_{limzl}VMIN + PIQ_{limzu}VMAX - PIQ_y\right)$
Vsel_x	$Vsel_x$	Algeb	$PIQ_ySWV_{s1} + SWV_{s0}\left(QcpfSWPF_{s1} + QrefSWPF_{s0} + Vref_1\right) - Vsel_x$
Verr	$Verr$	Algeb	$-Verr + Vref0r - s_{0y}$
dbV_y	dbV_y	Algeb	$1.0dbV_{dbzl}\left(Verr - dbd_1\right) + 1.0dbV_{dbzu}\left(Verr - dbd_2\right) - dbV_y$
Iqinj	$Iqinj$	Algeb	$-Iqinj + KqvVolt_{dip}dbV_y + fThld\left(1 - Volt_{dip}\right)\left(IqfrzpThld + KqvdbV_ynThld\right)$
wg	wg	Algeb	$1.0 - wg$
Pref	$Pref$	Algeb	$-Kf\left(\omega - 1\right) - Pref + \frac{Pref_0}{wg}$
Psel	$Psel$	Algeb	$-Psel + SWP_{s0}pfilt_y + SWP_{s1}pfilt_ywg$
VDL1_y	VDL_{1y}	Algeb	$-VDL_{1y} + \text{FixPiecewise}\left(\left(Iq_1,\ Vq_1 \geq s_{0y}\right), \left(Iq_1 + kVq_{12}\left(-Vq_1 + s_{0y}\right),\ Vq_2 \geq s_0\right.\right.$
VDL2_y	VDL_{2y}	Algeb	$-VDL_{2y} + \text{FixPiecewise}\left(\left(Ip_1,\ Vp_1 \geq s_{0y}\right), \left(Ip_1 + kVp_{12}\left(-Vp_1 + s_{0y}\right),\ Vp_2 \geq s\right.\right.$
Ipmax	$Ipmax$	Algeb	$-Ipmax + IpmaxhfThld_2 + \left(1 - fThld_2\right)\left(\sqrt{Ipmax2sq}SWPQ_{s0} + SWPQ_{s1}\left(z\right.\right.$
Iqmax	$Iqmax$	Algeb	$-Iqmax + \sqrt{Iqmax2sq}SWPQ_{s1} + SWPQ_{s0}\left(zVDL_1\left(Imaxr\left(1 - VDL1c\right) + V\right.\right.$
PIV_ys	PIV_{ys}	Algeb	$\left(1 - Volt_{dip}\right)\left(KvpSWQ_{s1}\left(-SWV_{s0}s_{0y} + VMAXVsel_{limzu} + VMINVsel_{limzl} +\right.\right.$
PIV_y	PIV_y	Algeb	$\left(1 - Volt_{dip}\right)\left(IqmaxPIV_{limzu} + IqminPIV_{limzl} + PIV_{limzi}PIV_{ys} - PIV_y\right)$
Qsel	$Qsel$	Algeb	$PIV_ySWQ_{s1} - Qsel + SWQ_{s0}s_{4y}$
IpHL_x	$IpHL_x$	Algeb	$-IpHL_x + \frac{s_{5y}}{vp}$
IqHL_x	$IqHL_x$	Algeb	$-IqHL_x + Iqinj + Qsel$

Table 23 – continued from p

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
a	a	ExtAlgeb	0
v	v	ExtAlgeb	0
Pe	Pe	ExtAlgeb	0
Qe	Qe	ExtAlgeb	0
Ipcmd	$Ipcmd$	ExtAlgeb	$ue\left(IpHL_{limzi}IpHL_x + IpHL_{limzl}Ipmin + IpHL_{limzu}Ipmax - Ipcmd_0\right)$
Iqcmd	$Iqcmd$	ExtAlgeb	$ue\left(-IqHL_{limzi}IqHL_x - IqHL_{limzl}Iqmin - IqHL_{limzu}Iqmax - Iqcmd_0\right)$

Services

Name	Symbol	Equation
Ipcmd0	$Ipcmd0$	$\frac{p_0}{v}$
Iqcmd0	$Iqcmd0$	$-\frac{q_0}{v}$
pfaref0	Φ_{ref0}	$\text{atan}_2\left(q_0, p_0\right)$
Volt_dip	z_{Vdip}	$1 - Vcmp_{zi}$
qref0	q_{ref0}	$Iqcmd_0 SWQ_{s0}\left(VLower_{zi}v + 0.01VLower_{zl}\right) + SWQ_{s1}\left(-Vref_1 + v\right)$
PIQ_flag	z_{PIQ}^{flag}	0
s4_flag	$z_{s_4}^{flag}$	0
Vref0r	$V_{ref0,r}$	$Vref_0$
pThld	p_{Thld}	Indicator $\left(Thld > 0\right)$
nThld	n_{Thld}	Indicator $\left(Thld < 0\right)$
Thld_abs	$\lvert Thld\rvert$	$\lvert Thld\rvert$
fThld	$fThld$	0
Pref0	P_{ref0}	p_0
s5_flag	z_{s5}^{flag}	0
kVq12	k_{Vq12}	$\frac{-Iq_1+Iq_2}{-Vq_1+Vq_2}$
kVq23	k_{Vq23}	$\frac{-Iq_2+Iq_3}{-Vq_2+Vq_3}$
kVq34	k_{Vq34}	$\frac{-Iq_3+Iq_4}{-Vq_3+Vq_4}$
zVDL1	z_{VDL1}	$Iq_1 \leq Iq_2 \wedge Iq_2 \leq Iq_3 \wedge Iq_3 \leq Iq_4 \wedge Vq_1 \leq Vq_2 \wedge Vq_2 \leq Vq_3 \wedge Vq_3 \leq Vq_4$
kVp12	k_{Vp12}	$\frac{-Ip_1+Ip_2}{-Vp_1+Vp_2}$
kVp23	k_{Vp23}	$\frac{-Ip_2+Ip_3}{-Vp_2+Vp_3}$
kVp34	k_{Vp34}	$\frac{-Ip_3+Ip_4}{-Vp_3+Vp_4}$
zVDL2	z_{VDL2}	$Ip_1 \leq Ip_2 \wedge Ip_2 \leq Ip_3 \wedge Ip_3 \leq Ip_4 \wedge Vp_1 \leq Vp_2 \wedge Vp_2 \leq Vp_3 \wedge Vp_3 \leq Vp_4$
fThld2	$fThld2$	0
VDL1c	$VDL1c$	$VDL_{1y} < Imaxr$
VDL2c	$VDL2c$	$VDL_{2y} < Imaxr$
Ipmax2sq0	$I_{pmax20,nn}^2$	FixPiecewise $\left(\left(0,\ Imax^2 - Iqcmd_0^2 \leq 0\right), \left(Imax^2 - Iqcmd_0^2,\ \text{True}\right)\right)$
Ipmax2sq	I_{pmax2}^2	FixPiecewise $\Big(\Big(0,\ Imax^2 - \left(IqHL_{limzi}IqHL_x + IqHL_{limzl}Iqmin + IqHL_{limzu}Iqmax\right)^2$
Ipmaxh	$Ipmaxh$	0
Iqmax2sq0	$I_{qmax,nn}^2$	FixPiecewise $\left(\left(0,\ Imax^2 - Ipcmd_0^2 \leq 0\right), \left(Imax^2 - Ipcmd_0^2,\ \text{True}\right)\right)$

Table 24 – continued fro

Name	Symbol	Equation
Iqmax2sq	I_{qmax2}^2	FixPiecewise $\left(\left(0,\ Imax^2 - (IpHL_{limzi}IpHL_x + IpHL_{limzl}Ipmin + IpHL_{limzu}Ipmax)^2\right.\right.$
Ipmin	I_{pmin}	0.0
PIV_flag	z_{PIV}^{flag}	0

Discretes

Name	Symbol	Type	Info
SWPF	SW_{PF}	Switcher	
SWV	SW_V	Switcher	
SWQ	SW_V	Switcher	
SWP	SW_P	Switcher	
SWPQ	SW_{PQ}	Switcher	
zp	zp	IsEqual	
Vcmp	V_{cmp}	Limiter	Voltage dip comparator
VLower	$VLower$	Limiter	Limiter for lower voltage cap
PFlim	$PFlim$	Limiter	
PIQ_lim	lim_{PIQ}	HardLimiter	
Vsel_lim	lim_{Vsel}	HardLimiter	
dbV_db	db_{dbV}	DeadBand	
pfilt_lim	$lim_{P_{filt}}$	RateLimiter	Rate limiter in Lag
s5_lim	lim_{s5}	AntiWindup	Limiter in Lag
PIV_lim	lim_{PIV}	HardLimiter	
IpHL_lim	lim_{IpHL}	HardLimiter	
IqHL_lim	lim_{IqHL}	HardLimiter	

Blocks

Name	Symbol	Type	Info
s0	$s0$	Lag	Voltage filter
S1	S_1	Lag	Pe filter
PIQ	PIQ	PITrackAWFreeze	
Vsel	$Vsel$	GainLimiter	Selection output of VFLAG
s4	s_4	LagFreeze	Filter for calculated voltage with freeze
dbV	dbV	DeadBand1	Deadband for voltage error (ref0)
pfilt	P_{filt}	LagRate	Active power filter with rate limits
s5	$s5$	LagAWFreeze	
VDL1	V_{DL1}	Piecewise	Piecewise linear characteristics of Vq-Iq
VDL2	V_{DL2}	Piecewise	Piecewise linear characteristics of Vp-Ip
PIV	PIV	PITrackAWFreeze	
IpHL	$IpHL$	GainLimiter	
IqHL	$IqHL$	GainLimiter	

Setpoints

Name	Target	Info
pref	Pref0	initial P ref
qref	qref0	

Config Fields in [REECA1G]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)
kqs	K_{qs}	2	Q PI controller tracking gain	
kvs	K_{vs}	2	Voltage PI controller tracking gain	
tpfilt	T_{pfilt}	0.020	Time const. for Pref filter	

REECB1

Renewable energy electrical control model B (REEC_B).

REECB1 is a simplified variant of REECA1 with the following differences:

- No voltage-dependent current limit (VDL) piecewise characteristics. Current limits use simple algebraic expressions based on `Imax` and `PQFLAG` priority.
- No state transition logic for reactive current injection hold. `Iqinj` is a direct proportional gain with limits.
- No `PFLAG` speed dependency. `Pref` feeds directly through the rate limiter to `Pord`.
- No `Vref1`. Only `Vref0` is used.
- `QFLAG=0` uses direct reactive current (`PFsel/vp`) via a lag filter, bypassing the voltage PI controller.

Regarding the reactive current injection during voltage dip:

- Exercise caution when coordinating `dbd1`, `dbd2`, `Vdip`, and `Vup` to avoid unintended responses.
- `Kqv` controls the intensity of reactive power injection and needs to be tuned properly to avoid voltage overshoot.

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
reg		Renewable generator idx			mandatory
busr		Optional remote bus for voltage control			
PFFLAG		Power factor control flag; 1-PF control, 0-Q control		*bool*	mandatory
VFLAG		Voltage control flag; 1-Q control, 0-V control		*bool*	mandatory
QFLAG		Q control flag; 1-V or Q control, 0-const. PF or Q		*bool*	mandatory
PQFLAG		P/Q priority flag for I limit; 0-Q priority, 1-P priority		*bool*	mandatory
Vdip	V_{dip}	Low V threshold to activate Iqinj logic	0.800	*p.u.*	
Vup	V_{up}	V threshold above which to activate Iqinj logic	1.200	*p.u.*	
Trv	T_{rv}	Voltage filter time constant	0.020		
dbd1	d_{bd1}	Lower bound of the voltage deadband (<=0)	-0.020		
dbd2	d_{bd2}	Upper bound of the voltage deadband (>=0)	0.020		
Kqv	K_{qv}	Gain to compute Iqinj from V error (caution!!)	1		
Iqh1	I_{qh1}	Upper limit on Iqinj	999		
Iql1	I_{ql1}	Lower limit on Iqinj	-999		
Vref0	V_{ref0}	User defined Vref (if 0, use initial bus V)	1		
Tp	T_p	Filter time constant for Pe	0.020	*s*	
QMax	Q_{max}	Upper limit for reactive power regulator	999		power
QMin	Q_{min}	Lower limit for reactive power regulator	-999		power

continues on next page

Table 25 – continued from previous page

Name	Symbol	Description	Default	Unit	Properties
VMAX	V_{max}	Upper limit for voltage control	999		
VMIN	V_{min}	Lower limit for voltage control	-999		
Kqp	K_{qp}	Proportional gain for reactive power error	1		
Kqi	K_{qi}	Integral gain for reactive power error	0.100		
Kvp	K_{vp}	Proportional gain for voltage error	1		
Kvi	K_{vi}	Integral gain for voltage error	0.100		
Tiq	T_{iq}	Filter time constant for Iq (used when QFLAG=0)	0.020		
dPmax	d_{Pmax}	Power reference max. ramp rate (>0)	999		power
dPmin	d_{Pin}	Power reference min. ramp rate (<0)	-999		power
PMAX	P_{max}	Max. active power limit > 0	999		power
PMIN	P_{min}	Min. active power limit	0		power
Imax	I_{max}	Max. apparent current limit	999		current
Tpord	T_{pord}	Filter time constant for power setpoint	0.020		
ue	u_e	effective online status	0		
bus		Retrieved bus idx			
gen		Retrieved StaticGen idx			
Sn	S_n		0		

Variables

Name	Symbol	Type	Description	Unit	Properties
s0_y	s_{0y}	State	State in lag transfer function		v_str
S1_y	S_{1y}	State	State in lag transfer function		v_str
PIQ_xi	PIQ_{ξ}	State	Integrator output		v_str
PIV_xi	PIV_{ξ}	State	Integrator output		v_str
s4_y	s_{4y}	State	State in lag transfer function		v_str
pfilt_y	$pfilt_y$	State	State in lag TF		v_str
s5_y	s_{5y}	State	State in lag TF		v_str
Pord	$Pord$	AliasState	Alias of s5_y		
vp	vp	Algeb	Sensed lower-capped voltage		v_str
pfaref	$pfaref$	Algeb	power factor angle ref	*rad*	v_str
Qref	$Qref$	Algeb	external Q ref	*p.u.*	v_str
Qcpf	$Qcpf$	Algeb	Q calculated from P and power factor	*p.u.*	v_str
PFsel	$PFsel$	Algeb	Output of PFFLAG selector		v_str
Qerr	$Qerr$	Algeb	Reactive power error		v_str
PIQ_ys	PIQ_{ys}	Algeb	PI summation before limit		v_str
PIQ_y	PIQ_y	Algeb	PI output		v_str
Vsel_x	$Vsel_x$	Algeb	Value before limiter		v_str
Verr	$Verr$	Algeb	Voltage error (Vref0)		v_str
dbV_y	dbV_y	Algeb	Deadband type 1 output		v_str
Iqinj_x	$Iqinj_x$	Algeb	Value before limiter		v_str

continues on next page

Table 26 – continued from previous page

Name	Symbol	Type	Description	Unit	Properties
Ipmax	$Ipmax$	Algeb	Upper limit on Ipcmd		v_str
Iqmax	$Iqmax$	Algeb	Upper limit on Iqcmd		v_str
PIV_ys	PIV_{ys}	Algeb	PI summation before limit		v_str
PIV_y	PIV_y	Algeb	PI output		v_str
Qsel	$Qsel$	Algeb	Selection output of QFLAG		v_str
IpHL_x	$IpHL_x$	Algeb	Value before limiter		v_str
IqHL_x	$IqHL_x$	Algeb	Value before limiter		v_str
wg	wg	Algeb	Drive train generator speed		v_str
Pref	$Pref$	Algeb	external P ref	*p.u.*	v_str
a	a	ExtAlgeb	Bus voltage angle		
v	v	ExtAlgeb	d-axis bus voltage magnitude		
Pe	Pe	ExtAlgeb	Retrieved Pe of RenGen		
Qe	Qe	ExtAlgeb	Retrieved Qe of RenGen		
Ipcmd	$Ipcmd$	ExtAlgeb	Retrieved Ipcmd of RenGen		
Iqcmd	$Iqcmd$	ExtAlgeb	Retrieved Iqcmd of RenGen		

Initialization Equations

Name	Symbol	Type	Initial Value
s0_y	s_{0y}	State	v
S1_y	S_{1y}	State	Pe
PIQ_xi	PIQ_ξ	State	0.0
PIV_xi	PIV_ξ	State	$-Iqcmd_0 SWQ_{s1}$
s4_y	s_{4y}	State	$\frac{PFsel}{vp}$
pfilt_y	$pfilt_y$	State	$Pref$
s5_y	s_{5y}	State	$pfilt_y$
Pord	$Pord$	AliasState	
vp	vp	Algeb	$VLower_{zi} v + 0.01 VLower_{zl}$
pfaref	$pfaref$	Algeb	$pfaref_0$
Qref	$Qref$	Algeb	$-qref_0$
Qcpf	$Qcpf$	Algeb	q_0
PFsel	$PFsel$	Algeb	$QcpfSWPF_{s1} + QrefSWPF_{s0}$
Qerr	$Qerr$	Algeb	$PFlim_{zi} PFsel + PFlim_{zl} QMin + PFlim_{zu} QMax - Qe$
PIQ_ys	PIQ_{ys}	Algeb	$KqpQerrSWV_{s1}$
PIQ_y	PIQ_y	Algeb	$PIQ_{limzi} PIQ_{ys} + PIQ_{limzl} VMIN + PIQ_{limzu} VMAX$
Vsel_x	$Vsel_x$	Algeb	$PIQ_y SWV_{s1}$
Verr	$Verr$	Algeb	$Vref0r - s_{0y}$
dbV_y	dbV_y	Algeb	$1.0 dbV_{dbzl} (Verr - dbd_1) + 1.0 dbV_{dbzu} (Verr - dbd_2)$
Iqinj_x	$Iqinj_x$	Algeb	$KqvdbV_y$
Ipmax	$Ipmax$	Algeb	$ImaxrSWPQ_{s1} + \sqrt{Ipmax2sq0} SWPQ_{s0}$
Iqmax	$Iqmax$	Algeb	$ImaxrSWPQ_{s0} + \sqrt{Iqmax2sq0} SWPQ_{s1}$

continues on ne

Table 27 – continued from previous page

Name	Symbol	Type	Initial Value
PIV_ys	PIV_{ys}	Algeb	$-Iqcmd_0SWQ_{s1} + KvpSWQ_{s1}\left(VMAXVsel_{limzu} + VMINVsel_{limzl} + Vsel_{limz}\right.$
PIV_y	PIV_y	Algeb	$IqmaxPIV_{limzu} + IqminPIV_{limzl} + PIV_{limzi}PIV_{ys}$
Qsel	$Qsel$	Algeb	$PIV_ySWQ_{s1} + SWQ_{s0}s_{4y}$
IpHL_x	$IpHL_x$	Algeb	$\frac{s_{5y}}{vp}$
IqHL_x	$IqHL_x$	Algeb	$Iqh_1Iqinj_{limzu} + Iqinj_{limzi}Iqinj_x + Iqinj_{limzl}Iql_1 + Qsel$
wg	wg	Algeb	1.0
Pref	$Pref$	Algeb	p_0
a	a	ExtAlgeb	
v	v	ExtAlgeb	
Pe	Pe	ExtAlgeb	
Qe	Qe	ExtAlgeb	
Ipcmd	$Ipcmd$	ExtAlgeb	
Iqcmd	$Iqcmd$	ExtAlgeb	

Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
s0_y	s_{0y}	State	$-s_{0y} + v$	T_{rv}
S1_y	S_{1y}	State	$Pe - S_{1y}$	T_p
PIQ_x	PIQ_ξ	State	$Kqi\left(1 - Volt_{dip}\right)\left(2PIQ_y - 2PIQ_{ys} + QerrSWV_{s1}\right)$	
PIV_x	PIV_ξ	State	$Kvi\left(1 - Volt_{dip}\right)\left(2PIV_y - 2PIV_{ys} + SWQ_{s1}\left(VMAXVsel_{limzu} + \right.\right.$	$Vsel_{limzl} + Vse$
s4_y	s_{4y}	State	$\left(1 - Volt_{dip}\right)\left(\frac{PFsel}{vp} - s_{4y}\right)$	T_{iq}
pfilt_y	$pfilt_y$	State	$Pref - pfilt_y$	0.02
s5_y	s_{5y}	State	$\left(1 - Volt_{dip}\right)\left(pfilt_y - s_{5y}\right)$	T_{pord}
Pord	$Pord$	AliasSta	0	

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
vp	vp	Algeb	$VLower_{zi}v + 0.01VLower_{zl} - vp$
pfaref	$pfaref$	Algeb	$-pfaref + pfaref_0$
Qref	$Qref$	Algeb	$-Qref - qref_0$
Qcpf	$Qcpf$	Algeb	$(1 - zp_{z1})\left(-Qcpf + S_{1y}\tan\left(pfaref\right)\right)$
PFsel	$PFsel$	Algeb	$-PFsel + QcpfSWPF_{s1} + QrefSWPF_{s0}$
Qerr	$Qerr$	Algeb	$PFlim_{zi}PFsel + PFlim_{zl}QMin + PFlim_{zu}QMax - Qe - Qerr$
PIQ_ys	PIQ_{ys}	Algeb	$(1 - Volt_{dip})\left(KqpQerrSWV_{s1} + PIQ_{\xi} - PIQ_{ys}\right)$
PIQ_y	PIQ_y	Algeb	$(1 - Volt_{dip})\left(PIQ_{limzi}PIQ_{ys} + PIQ_{limzl}VMIN + PIQ_{limzu}VMAX - PIQ_y\right)$
Vsel_x	$Vsel_x$	Algeb	$PIQ_ySWV_{s1} - Vsel_x$
Verr	$Verr$	Algeb	$-Verr + Vref0r - s_{0y}$
dbV_y	dbV_y	Algeb	$1.0dbV_{dbzl}\left(Verr - dbd_1\right) + 1.0dbV_{dbzu}\left(Verr - dbd_2\right) - dbV_y$
Iqinj_x	$Iqinj_x$	Algeb	$-Iqinj_x + KqvdbV_y$
Ipmax	$Ipmax$	Algeb	$ImaxrSWPQ_{s1} - Ipmax + \sqrt{Ipmax2sq}SWPQ_{s0}$
Iqmax	$Iqmax$	Algeb	$ImaxrSWPQ_{s0} - Iqmax + \sqrt{Iqmax2sq}SWPQ_{s1}$
PIV_ys	PIV_{ys}	Algeb	$(1 - Volt_{dip})\left(KvpSWQ_{s1}\left(VMAXVsel_{limzu} + VMINVsel_{limzl} + Vsel_{limzi}Vsel_x\right) + PI\right.$
PIV_y	PIV_y	Algeb	$(1 - Volt_{dip})\left(IqmaxPIV_{limzu} + IqminPIV_{limzl} + PIV_{limzi}PIV_{ys} - PIV_y\right)$
Qsel	$Qsel$	Algeb	$PIV_ySWQ_{s1} - Qsel + SWQ_{s0}s_{4y}$
IpHL_x	$IpHL_x$	Algeb	$-IpHL_x + \frac{s_{5y}}{vp}$
IqHL_x	$IqHL_x$	Algeb	$-IqHL_x + Iqh_1Iqinj_{limzu} + Iqinj_{limzi}Iqinj_x + Iqinj_{limzl}Iql_1 + Qsel$
wg	wg	Algeb	$1.0 - wg$
Pref	$Pref$	Algeb	$-Pref + Pref_0$
a	a	ExtAlgeb	0
v	v	ExtAlgeb	0
Pe	Pe	ExtAlgeb	0
Qe	Qe	ExtAlgeb	0
Ipcmd	$Ipcmd$	ExtAlgeb	$ue\left(IpHL_{limzi}IpHL_x + IpHL_{limzl}Ipmin + IpHL_{limzu}Ipmax - Ipcmd_0\right)$
Iqcmd	$Iqcmd$	ExtAlgeb	$ue\left(-IqHL_{limzi}IqHL_x - IqHL_{limzl}Iqmin - IqHL_{limzu}Iqmax - Iqcmd_0\right)$

Services

Name	Symbol	Equation	Type
Ipcmd	$Ipcm$	$\frac{p_0}{v}$	ConstService
Iqcmd	$Iqcm$	$-\frac{q_0}{v}$	ConstService
pfaref	Φ_{ref0}	$\text{atan}_2\left(q_0, p_0\right)$	ConstService
Volt_	z_{Vdip}	$1 - Vcmp_{zi}$	VarService
qref0	q_{ref0}	$-q_0$	ConstService
PIQ_f	z_{PIQ}^{flag}	0	EventFlag
Vref0	$V_{ref0,}$	$Vref_0$	ConstService
Ipmax2	I_{pmax}^2	$\text{FixPiecewise}\left(\left(0,\ Imaxr^2 - Iqcmd_0^2 \leq 0\right), \left(Imaxr^2 - Iqcmd_0^2,\ \text{True}\right)\right)$	ConstService
Ipmax2	I_{pmax}^2	$\text{FixPiecewise}\left(\left(0,\ Imaxr^2 - (IqHL_{limzi}IqHL_x + IqHL_{limzl}Iqmin + IqHL_{li}\right.\right.$	VarSer- $ax)^2 \leq 0\Big), \Big(Im$ vice
Iqmax2	I_{qmax}^2	$\text{FixPiecewise}\left(\left(0,\ Imaxr^2 - Ipcmd_0^2 \leq 0\right), \left(Imaxr^2 - Ipcmd_0^2,\ \text{True}\right)\right)$	ConstService
Iqmax2	I_{qmax}^2	$\text{FixPiecewise}\left(\left(0,\ Imaxr^2 - (IpHL_{limzi}IpHL_x + IpHL_{limzl}Ipmin + IpHL_{li}\right.\right.$	VarSer- $ax)^2 \leq 0\Big), \Big(Im$ vice
Ipmin	I_{pmin}	0.0	ConstService
PIV_f	z_{PIV}^{flag}	0	EventFlag
s4_fla	$z_{s_4}^{flag}$	0	EventFlag
Pref0	P_{ref0}	p_0	ConstSer-

Discretes

Name	Symbol	Type	Info
SWPF	SW_{PF}	Switcher	
SWV	SW_V	Switcher	
SWQ	SW_Q	Switcher	
SWPQ	SW_{PQ}	Switcher	
zp	zp	IsEqual	
Vcmp	V_{cmp}	Limiter	Voltage dip comparator
VLower	$VLower$	Limiter	Limiter for lower voltage cap
PFlim	$PFlim$	Limiter	
PIQ_lim	lim_{PIQ}	HardLimiter	
Vsel_lim	lim_{Vsel}	HardLimiter	
dbV_db	db_{dbV}	DeadBand	
Iqinj_lim	lim_{Iqinj}	HardLimiter	
PIV_lim	lim_{PIV}	HardLimiter	
IpHL_lim	lim_{IpHL}	HardLimiter	
IqHL_lim	lim_{IqHL}	HardLimiter	
pfilt_lim	$lim_{P_{filt}}$	RateLimiter	Rate limiter in Lag
s5_lim	lim_{s5}	AntiWindup	Limiter in Lag

Blocks

Name	Symbol	Type	Info
s0	$s0$	Lag	Voltage filter
S1	S_1	Lag	Pe filter
PIQ	PIQ	PITrackAWFreeze	
Vsel	$Vsel$	GainLimiter	Selection output of VFLAG
dbV	dbV	DeadBand1	Deadband for voltage error (ref0)
Iqinj	$Iqinj$	GainLimiter	Gain-limited Iqinj from voltage error
PIV	PIV	PITrackAWFreeze	
s4	s_4	LagFreeze	Filter for reactive current (QFLAG=0 path)
IpHL	$IpHL$	GainLimiter	
IqHL	$IqHL$	GainLimiter	
pfilt	P_{filt}	LagRate	Active power filter with rate limits
s5	$s5$	LagAWFreeze	

Setpoints

Name	Target	Info
pref	Pref0	initial P ref
qref	qref0	

Config Fields in [REECB1]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)
kqs	K_{qs}	2	Q PI controller tracking gain	
kvs	K_{vs}	2	Voltage PI controller tracking gain	
tpfilt	T_{pfilt}	0.020	Time const. for Pref filter	

5.3.25 RenGen

Renewable generator (converter) group.

See *SynGen* for the notes on replacing StaticGen and setting the power ratio parameters.

Attention is needed for the power base `Sn`. When replacing a synchronous generator, the renewable generator should have the same or larger `Sn`. Improper `Sn` will cause the initial values to exceed typical per-unit ranges and cause the initialization to fail.

Common Parameters: u, name, bus, gen, Sn

Common Variables: Pe, Qe

Available models: *REGCA1*, *REGCP1*, *REGCV1*, *REGCV2*, *REGF1*, *REGF2*, *REGF3*

REGCA1

Renewable energy generator model type A.

Implements `REGCA1` in PSS/E, or `REGC_A` in PSLF and Powerworld.

Volim is the voltage limit for high voltage reactive current management, which should be large than static bus voltage (Volim > v), or initialization error will occur.

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
bus		interface bus id			mandatory
gen		static generator index			mandatory
Sn	S_n	Model MVA base	100	*MVA*	
Tg	T_g	converter time const.	0.100	*s*	
Rrpwr	R_{rpwr}	Low voltage power logic (LVPL) ramp limit	10	*p.u.*	
Brkpt	B_{rkpt}	LVPL characteristic voltage 2	1	*p.u.*	
Zerox	Z_{erox}	LVPL characteristic voltage 1	0.500	*p.u*	
Lvplsw	z_{Lvplsw}	Low volt. P logic: 1-enable, 0-disable	1	*bool*	
Lvpl1	L_{vpl1}	LVPL gain at Brkpt	1	*p.u*	
Volim	V_{olim}	Voltage lim for high volt. reactive current mgnt.	1.200	*p.u.*	
Lvpnt1	L_{vpnt1}	High volt. point for low volt. active current mgnt.	0.800	*p.u.*	
Lvpnt0	L_{vpnt0}	Low volt. point for low volt. active current mgnt.	0.400	*p.u.*	
Iolim	I_{olim}	lower current limit for high volt. reactive current mgnt.	-1.500	*p.u. (mach base)*	current
Tfltr	T_{fltr}	Voltage filter T const for low volt. active current mgnt.	0.100	*s*	
Khv	K_{hv}	Overvolt. compensation gain in high volt. reactive current mgnt.	0.700		
Iqrmax	I_{qrmax}	Upper limit on the ROC for reactive current	1	*p.u.*	current
Iqrmin	I_{qrmin}	Lower limit on the ROC for reactive current	-1	*p.u.*	current
Accel	A_{ccel}	Acceleration factor	0		
gammap	γ_P	P ratio of linked static gen	1		
gammaq	γ_Q	Q ratio of linked static gen	1		
ue	u_e	effective online status	0		
ra	r_a		0		
xs	x_s		0		

Variables

Name	Symbol	Type	Description	Unit	Properties
S1_y	S_{1y}	State	State in lag TF		v_str
S2_y	S_{2y}	State	State in lag transfer function		v_str
S0_y	S_{0y}	State	State in lag TF		v_str
LVG_y	LVG_y	Algeb	Output of piecewise		v_str
Ipcmd	$Ipcmd$	Algeb	current component for active power		v_str
Iqcmd	$Iqcmd$	Algeb	current component for reactive power		v_str
LVPL_y	$LVPL_y$	Algeb	Output of piecewise		v_str
Ipout	$Ipout$	Algeb	Output Ip current		v_str
HVG_x	HVG_x	Algeb	Value before limiter		v_str
Iqout_x	$Iqout_x$	Algeb	Value before limiter		v_str
Pe	Pe	Algeb	Active power output		v_str
Qe	Qe	Algeb	Reactive power output		v_str
a	a	ExtAlgeb	Bus voltage angle		
v	v	ExtAlgeb	Bus voltage magnitude		
vd	vd	AliasAlgeb	Alias of v		

Initialization Equations

Name	Symbol	Type	Initial Value
S1_y	S_{1y}	State	$-Iqcmd$
S2_y	S_{2y}	State	$1.0v$
S0_y	S_{0y}	State	$Ipcmd$
LVG_	$LVG_$	Algeb	FixPiecewise $((0,\ Lvpnt_0 \geq v), (kLVG(-Lvpnt_0 + v),\ Lvpnt_1 \geq v), (1,\ \text{True}))$
Ipcmd	$Ipcm$	Algeb	$\frac{Ipcmd_0 \text{ Indicator } (LVG_y > 0)}{LVG_y} + \text{Indicator } (LVG_y \leq 0)$
Iqcmd	$Iqcm$	Algeb	$Iqcmd_0$
LVPL_	LVP	Algeb	FixPiecewise $((9999 - 9999Lvplsw,\ S_{2y} \leq Zerox), (-9999Lvplsw + kLVPL(S_{2y} - Zerox$
Ipout	$Ipout$	Algeb	$IpcmdLVG_y$
HVG_	HVG	Algeb	$Khv(-Volim + v)$
Iqout_	$Iqout$	Algeb	$-HVG_{limzi}HVG_x + S_{1y}$
Pe	Pe	Algeb	p_0
Qe	Qe	Algeb	q_0
a	a	ExtAlgeb	
v	v	ExtAlgeb	
vd	vd	AliasAlgeb	

Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
S1_y	S_{1y}	State	$-Iqcmd - S_{1y}$	T_g
S2_y	S_{2y}	State	$-S_{2y} + 1.0v$	T_{fltr}
S0_y	S_{0y}	State	$Ipcmd - S_{0y}$	T_g

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
LVG_	LVG	Algeb	$-LVG_y$+FixPiecewise $((0,\ Lvpnt_0 \geq v), (kLVG(-Lvpnt_0 + v),\ Lvpnt_1 \geq v), (1,\ \text{True}))$
Ipcmd	$Ipcm$	Algeb	$-Ipcmd + Ipcmd_{0LVG}$
Iqcmd	$Iqcm$	Algeb	$-Iqcmd + Iqcmd_0$
LVPL	LVP	Algeb	$-LVPL_y$+FixPiecewise $((9999 - 9999Lvplsw,\ S_{2y} \leq Zerox), (-9999Lvplsw + kLVPL(S_{2y}$
Ipout	$Ipout$	Algeb	$-Ipout + LVG_y S_{0y}$
HVG_	HVG	Algeb	$-HVG_x + Khv(-Volim + v)$
Iqout_	$Iqout$	Algeb	$-HVG_{limzi} HVG_x - Iqout_x + S_{1y}$
Pe	Pe	Algeb	$Ipoutv - Pe$
Qe	Qe	Algeb	$-Qe + v(IolimIqout_{limzl} + Iqout_{limzi} Iqout_x)$
a	a	ExtAlgeb	$-Peue$
v	v	ExtAlgeb	$-Qeue$
vd	vd	AliasAlgeb	0

Services

Name	Symbol	Equation	Type
p0	P_0	$gammapp0s$	ConstService
q0	Q_0	$gammaqq0s$	ConstService
q0gt0	$z_{q0>0}$	Indicator $(q_0 > 0)$	ConstService
q0lt0	$z_{q0<0}$	Indicator $(q_0 < 0)$	ConstService
Ipcmd0	I_{pcmd0}	$\frac{p_0}{v}$	ConstService
Iqcmd0	I_{qcmd0}	$-\frac{q_0}{v}$	ConstService
Ipcmd0_LVG	$Ipcmd0_LVG$	$Ipcmd$	PostInitService
kLVG	k_{LVG}	$\frac{1}{-Lvpnt_0+Lvpnt_1}$	ConstService
kLVPL	k_{LVPL}	$\frac{Lvpl_1 Lvplsw}{Brkpt-Zerox}$	ConstService

Discretes

Name	Symbol	Type	Info
S1_lim	lim_{S_1}	AntiWindupRate	Limiter in Lag
S0_lim	lim_{S_0}	AntiWindupRate	Limiter in Lag
HVG_lim	$lim_{H_{VG}}$	HardLimiter	
Iqout_lim	$lim_{I^{qout}}$	HardLimiter	

Blocks

Name	Symbol	Type	Info
LVG	L_{VG}	Piecewise	Ip gain during low voltage
S1	S_1	LagAntiWindupRate	Iqcmd delay
S2	S_2	Lag	Voltage filter with no anti-windup
LVPL	L_{VPL}	Piecewise	Low voltage Ipcmd upper limit
S0	S_0	LagAntiWindupRate	
HVG	H_{VG}	GainLimiter	High voltage gain block
Iqout	I^{qout}	GainLimiter	Iq output block

Config Fields in [REGCA1]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

REGCP1

Renewable energy generator model (REGC_A) with PLL.

A PLL device can be specified for estimating the phase angle at the coupling bus through the `pll` parameter:

- If `pll` is not given, the accurate bus angle will be used.
- If `pll` is not a valid PLL device, the program will error out.
- The program does not check if the provided `pll` actually measures the bus on which the converter is at.

One needs to carefully tune the PLL parameters to match the desired performance.

All other remarks for `REGCA1` apply.

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
bus		interface bus id			mandatory
gen		static generator index			mandatory
Sn	S_n	Model MVA base	100	*MVA*	
Tg	T_g	converter time const.	0.100	*s*	
Rrpwr	R_{rpwr}	Low voltage power logic (LVPL) ramp limit	10	*p.u.*	
Brkpt	B_{rkpt}	LVPL characteristic voltage 2	1	*p.u.*	
Zerox	Z_{erox}	LVPL characteristic voltage 1	0.500	*p.u*	
Lvplsw	z_{Lvplsw}	Low volt. P logic: 1-enable, 0-disable	1	*bool*	
Lvpl1	L_{vpl1}	LVPL gain at Brkpt	1	*p.u*	
Volim	V_{olim}	Voltage lim for high volt. reactive current mgnt.	1.200	*p.u.*	
Lvpnt1	L_{vpnt1}	High volt. point for low volt. active current mgnt.	0.800	*p.u.*	
Lvpnt0	L_{vpnt0}	Low volt. point for low volt. active current mgnt.	0.400	*p.u.*	
Iolim	I_{olim}	lower current limit for high volt. reactive current mgnt.	-1.500	*p.u. (mach base)*	current
Tfltr	T_{fltr}	Voltage filter T const for low volt. active current mgnt.	0.100	*s*	
Khv	K_{hv}	Overvolt. compensation gain in high volt. reactive current mgnt.	0.700		
Iqrmax	I_{qrmax}	Upper limit on the ROC for reactive current	1	*p.u.*	current
Iqrmin	I_{qrmin}	Lower limit on the ROC for reactive current	-1	*p.u.*	current
Accel	A_{ccel}	Acceleration factor	0		
gammap	γ_P	P ratio of linked static gen	1		
gammaq	γ_Q	Q ratio of linked static gen	1		
pll		Phase-lock loop device idx			
ue	u_e	effective online status	0		
ra	r_a		0		
xs	x_s		0		

Variables

Name	Symbol	Type	Description	Unit	Properties
S1_y	S_{1y}	State	State in lag TF		v_str
S2_y	S_{2y}	State	State in lag transfer function		v_str
S0_y	S_{0y}	State	State in lag TF		v_str
am	am	ExtState	Measured angle		
LVG_y	LVG_y	Algeb	Output of piecewise		v_str
Ipcmd	$Ipcmd$	Algeb	current component for active power		v_str
Iqcmd	$Iqcmd$	Algeb	current component for reactive power		v_str
LVPL_y	$LVPL_y$	Algeb	Output of piecewise		v_str
Ipout	$Ipout$	Algeb	Output Ip current		v_str
HVG_x	HVG_x	Algeb	Value before limiter		v_str
Iqout_x	$Iqout_x$	Algeb	Value before limiter		v_str
Pe	Pe	Algeb	Active power output		v_str
Qe	Qe	Algeb	Reactive power output		v_str
vd	vd	Algeb	d-axis voltage		v_str
vq	vq	Algeb	q-axis voltage		v_str
a	a	ExtAlgeb	Bus voltage angle		
v	v	ExtAlgeb	Bus voltage magnitude		

Initialization Equations

Name	Symbol	Type	Initial Value
S1_y	S_{1y}	State	$-Iqcmd$
S2_y	S_{2y}	State	$1.0v$
S0_y	S_{0y}	State	$Ipcmd$
am	am	ExtStat	
LVG_	LVG_y	Algeb	FixPiecewise $((0,\ Lvpnt_0 \geq v), (kLVG\,(-Lvpnt_0 + v),\ Lvpnt_1 \geq v), (1,\ \text{True}))$
Ipcmd	$Ipcmd$	Algeb	$\frac{Ipcmd_0 \text{ Indicator } (LVG_y > 0)}{LVG_y} + \text{Indicator } (LVG_y \leq 0)$
Iqcmd	$Iqcmd$	Algeb	$Iqcmd_0$
LVPL_	$LVPL$	Algeb	FixPiecewise $((9999 - 9999Lvplsw,\ S_{2y} \leq Zerox), (-9999Lvplsw + kLVPL\,(S_{2y} - Zerox)$
Ipout	$Ipout$	Algeb	$IpcmdLVG_y$
HVG_	HVG	Algeb	$Khv\,(-Volim + v)$
Iqout_	$Iqout$	Algeb	$-HVG_{limzi}HVG_x + S_{1y}$
Pe	Pe	Algeb	p_0
Qe	Qe	Algeb	q_0
vd	vd	Algeb	v
vq	vq	Algeb	0
a	a	ExtAlgeb	
v	v	ExtAlgeb	

Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
S1_y	S_{1y}	State	$-Iqcmd - S_{1y}$	T_g
S2_y	S_{2y}	State	$-S_{2y} + 1.0v$	T_{fltr}
S0_y	S_{0y}	State	$Ipcmd - S_{0y}$	T_g
am	am	ExtState	0	

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
LVG_	LVG	Algeb	$-LVG_y$+FixPiecewise $((0,\ Lvpnt_0 \geq v), (kLVG(-Lvpnt_0 + v),\ Lvpnt_1 \geq v), (1,\ \text{True}))$
Ipcmd	$Ipcm$	Algeb	$-Ipcmd + Ipcmd_{0LVG}$
Iqcmd	$Iqcm$	Algeb	$-Iqcmd + Iqcmd_0$
LVPL	LVP	Algeb	$-LVPL_y$+FixPiecewise $((9999 - 9999Lvplsw,\ S_{2y} \leq Zerox), (-9999Lvplsw + kLVPL(S_{2y}$
Ipout	$Ipout$	Algeb	$-Ipout + LVG_y S_{0y}$
HVG_	HVG	Algeb	$-HVG_x + Khv(-Volim + v)$
Iqout_	$Iqout$	Algeb	$-HVG_{limzi} HVG_x - Iqout_x + S_{1y}$
Pe	Pe	Algeb	$Ipoutvd - Pe + vq(IolimIqout_{limzl} + Iqout_{limzi} Iqout_x)$
Qe	Qe	Algeb	$-Ipoutvq - Qe + vd(IolimIqout_{limzl} + Iqout_{limzi} Iqout_x)$
vd	vd	Algeb	$v\cos(zam(a - am)) - vd$
vq	vq	Algeb	$-v\sin(zam(a - am)) - vq$
a	a	ExtAlgeb	$-Peue$
v	v	ExtAlgeb	$-Qeue$

Services

Name	Symbol	Equation	Type
p0	P_0	$gammapp0s$	ConstService
q0	Q_0	$gammaqq0s$	ConstService
q0gt0	$z_{q0>0}$	Indicator $(q_0 > 0)$	ConstService
q0lt0	$z_{q0<0}$	Indicator $(q_0 < 0)$	ConstService
Ipcmd0	I_{pcmd0}	$\frac{p_0}{v}$	ConstService
Iqcmd0	I_{qcmd0}	$-\frac{q_0}{v}$	ConstService
Ipcmd0_LVG	$Ipcmd0_LVG$	$Ipcmd$	PostInitService
kLVG	k_{LVG}	$\frac{1}{-Lvpnt_0+Lvpnt_1}$	ConstService
kLVPL	k_{LVPL}	$\frac{Lvpl_1 Lvplsw}{Brkpt-Zerox}$	ConstService

Discretes

Name	Symbol	Type	Info
S1_lim	lim_{S_1}	AntiWindupRate	Limiter in Lag
S0_lim	lim_{S_0}	AntiWindupRate	Limiter in Lag
HVG_lim	$lim_{H_{VG}}$	HardLimiter	
Iqout_lim	$lim_{I^{qout}}$	HardLimiter	

Blocks

Name	Symbol	Type	Info
LVG	L_{VG}	Piecewise	Ip gain during low voltage
S1	S_1	LagAntiWindupRate	Iqcmd delay
S2	S_2	Lag	Voltage filter with no anti-windup
LVPL	L_{VPL}	Piecewise	Low voltage Ipcmd upper limit
S0	S_0	LagAntiWindupRate	
HVG	H_{VG}	GainLimiter	High voltage gain block
Iqout	I^{qout}	GainLimiter	Iq output block

Config Fields in [REGCP1]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

REGCV1

Voltage-controlled VSC with VSG control.

Includes double-loop PI control and swing equation based VSG control. Voltage measurement delays are ignored.

Notes

- Extreme care needs to be taken when coordinating the PI controller parameters.
- Setting the primary frequency control droop `kw` can improve small-signal stability.
- The droop `kv` for voltage control (pu voltage / pu Q change), if used, needs to be chosen carefully. In most cases, `kv` should be a very small positive value if not zero.

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
bus		interface bus id			mandatory
gen		static generator index			mandatory
coi2		center of inertia 2 index			
Sn	S_n	Model MVA base	100	*MVA*	
fn	f	rated frequency	60		
Tc	T_c	switch time constant	0.010	*s*	
kw	k_ω	speed droop on active power (reciprocal of droop)	0	*p.u.*	non_negative,ipower
kv	k_v	reactive power droop on voltage	0	*p.u.*	non_negative,power
M	M	Emulated startup time constant (M=2H)	10	*s*	power
D	D	Emulated damping coefficient	0	*p.u.*	power
ra	r_a	resistance	0		z
xs	x_s	reactance	0.200		z
gammap	γ_P	P ratio of linked static gen	1		
gammaq	γ_Q	Q ratio of linked static gen	1		
Kpvd	kp_{vd}	vd controller proportional gain	0.500	*p.u.*	power
Kivd	ki_{vd}	vd controller integral gain	0.020	*p.u.*	power
Kpvq	kp_{vq}	vq controller proportional gain	0.500	*p.u.*	power
Kivq	ki_{vq}	vq controller integral gain	0.020	*p.u.*	power
KpId	kp_{di}	Id controller proportional gain	0.200	*p.u.*	power
KiId	ki_{di}	Id controller integral gain	0.010	*p.u.*	power
KpIq	kp_{qi}	Iq controller proportional gain	0.200	*p.u.*	power
KiIq	ki_{qi}	Iq controller integral gain	0.010	*p.u.*	power
ue	u_e	effective online status	0		

Variables

Name	Symbol	Type	Description	Unit	Properties
dw	dw	State	delta virtual rotor speed	*pu (Hz)*	v_str
delta	δ	State	virtual delta	*rad*	v_str
PIvd_xi	$PIvd_{\xi}$	State	Integrator output		v_str
PIvq_xi	$PIvq_{\xi}$	State	Integrator output		v_str
PIId_xi	$PIId_{\xi}$	State	Integrator output		v_str
PIIq_xi	$PIIq_{\xi}$	State	Integrator output		v_str
udLag_y	$udLag_y$	State	State in lag transfer function		v_str
uqLag_y	$uqLag_y$	State	State in lag transfer function		v_str
ud	ud	AliasState	Alias of udLag_y		
uq	uq	AliasState	Alias of uqLag_y		
Pref2	$Pref_2$	Algeb	active power reference after adjusting by frequency		v_str
vref2	$vref_2$	Algeb	voltage reference after adjusted by reactive power		v_str
omega	ω	Algeb	virtual rotor speed	*pu (Hz)*	v_str
vd	vd	Algeb	d-axis voltage		v_str
vq	vq	Algeb	q-axis voltage		v_str
Pe	Pe	Algeb	active power injection from VSC		v_str
Qe	Qe	Algeb	reactive power injection from VSC		v_str
Id	Id	Algeb	d-axis current		v_str
Iq	Iq	Algeb	q-axis current		v_str
PIvd_y	$PIvd_y$	Algeb	PI output		v_str
PIvq_y	$PIvq_y$	Algeb	PI output		v_str
PIId_y	$PIId_y$	Algeb	PI output		v_str
PIIq_y	$PIIq_y$	Algeb	PI output		v_str
udref	$udref$	Algeb	ud reference		v_str
uqref	$uqref$	Algeb	uq reference		v_str
a	a	ExtAlgeb	Bus voltage angle		
v	v	ExtAlgeb	Bus voltage magnitude		
Idref	$Idref$	AliasAlgeb	Alias of PIvd_y		
Iqref	$Iqref$	AliasAlgeb	Alias of PIvq_y		

Initialization Equations

Name	Symbol	Type	Initial Value
dw	dw	State	0
delta	δ	State	a
PIvd_xi	$PIvd_{\xi}$	State	Id_0
PIvq_xi	$PIvq_{\xi}$	State	Iq_0
PIId_xi	$PIId_{\xi}$	State	0.0
PIIq_xi	$PIIq_{\xi}$	State	0.0
udLag_y	$udLag_y$	State	$udref$
uqLag_y	$uqLag_y$	State	$uqref$
ud	ud	AliasState	
uq	uq	AliasState	
Pref2	$Pref_2$	Algeb	$Prefu$
vref2	$vref_2$	Algeb	$uvref$
omega	ω	Algeb	u
vd	vd	Algeb	vd_0
vq	vq	Algeb	vq_0
Pe	Pe	Algeb	$Pref$
Qe	Qe	Algeb	$Qref$
Id	Id	Algeb	Id_0
Iq	Iq	Algeb	Iq_0
PIvd_y	$PIvd_y$	Algeb	$Id_0 + Kpvd\,(vd - vref_2)$
PIvq_y	$PIvq_y$	Algeb	$Iq_0 + Kpvqvq$
PIId_y	$PIId_y$	Algeb	$KpId\,(Id - PIvd_y)$
PIIq_y	$PIIq_y$	Algeb	$KpIq\,(Iq - PIvq_y)$
udref	$udref$	Algeb	$udref_0$
uqref	$uqref$	Algeb	$uqref_0$
a	a	ExtAlgeb	
v	v	ExtAlgeb	
Idref	$Idref$	AliasAlgeb	
Iqref	$Iqref$	AliasAlgeb	

Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
dw	dw	State	$-Ddw - Pe + Pref_2$	M
delta	δ	State	$2\pi dwfn$	
PIvd_xi	$PIvd_\xi$	State	$Kivd\left(vd - vref_2\right)$	
PIvq_xi	$PIvq_\xi$	State	$Kivqvq$	
PIId_xi	$PIId_\xi$	State	$KiId\left(Id - PIvd_y\right)$	
PIIq_xi	$PIIq_\xi$	State	$KiIq\left(Iq - PIvq_y\right)$	
udLag_y	$udLag_y$	State	$-udLag_y + udref$	T_c
uqLag_y	$uqLag_y$	State	$-uqLag_y + uqref$	T_c
ud	ud	AliasState	0	
uq	uq	AliasState	0	

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
Pref2	$Pref_2$	Algeb	$Prefue - Pref_2 - dwkw$
vref2	$vref_2$	Algeb	$kv\left(-Qe + Qrefue\right) + vref - vref_2$
omega	ω	Algeb	$dw - \omega + 1$
vd	vd	Algeb	$uev\cos\left(a - \delta\right) - vd$
vq	vq	Algeb	$uev\sin\left(a - \delta\right) - vq$
Pe	Pe	Algeb	$Idvd + Iqvq - Pe$
Qe	Qe	Algeb	$Idvq - Iqvd - Qe$
Id	Id	Algeb	$Idra - Iqxs - udLag_y + vd$
Iq	Iq	Algeb	$Idxs + Iqra - uqLag_y + vq$
PIvd_y	$PIvd_y$	Algeb	$Kpvd\left(vd - vref_2\right) + PIvd_\xi - PIvd_y$
PIvq_y	$PIvq_y$	Algeb	$Kpvqvq + PIvq_\xi - PIvq_y$
PIId_y	$PIId_y$	Algeb	$KpId\left(Id - PIvd_y\right) + PIId_\xi - PIId_y$
PIIq_y	$PIIq_y$	Algeb	$KpIq\left(Iq - PIvq_y\right) + PIIq_\xi - PIIq_y$
udref	$udref$	Algeb	$-Iqrefxs + PIId_y - udref + vd$
uqref	$uqref$	Algeb	$Idrefxs + PIIq_y - uqref + vq$
a	a	ExtAlgeb	$-Peue$
v	v	ExtAlgeb	$-Qeue$
Idref	$Idref$	AliasAlgeb	0
Iqref	$Iqref$	AliasAlgeb	0

Services

Name	Symbol	Equation	Type
Pref	P_{ref}	$gammapp0s$	ConstService
Qref	Q_{ref}	$gammaqq0s$	ConstService
ixs	$1/xs$	$\frac{1}{xs}$	ConstService
Id0	I_{d0}	$\frac{Prefu}{v}$	ConstService
Iq0	I_{q0}	$-\frac{Qrefu}{v}$	ConstService
vd0	v_{d0}	uv	ConstService
vq0	v_{q0}	0	ConstService
udref0	u_{dref0}	$Id_0ra - Iq_0xs + vd_0$	ConstService
uqref0	u_{qref0}	$Id_0xs + Iq_0ra + vq_0$	ConstService

Blocks

Name	Symbol	Type	Info
PIvd	$PIvd$	PIController	
PIvq	$PIvq$	PIController	
PIId	$PIId$	PIController	
PIIq	$PIIq$	PIController	
udLag	$udLag$	Lag	
uqLag	$uqLag$	Lag	

Setpoints

Name	Target	Info
pref	Pref	Initial P for the REGCV1 device
qref	Qref	Initial Q for the REGCV1 device
vref	vref	initial v of the static gen

Config Fields in [REGCV1]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

REGCV2

Voltage-controlled VSC with VSG control.

The inner-loop current PI controllers are replaced with lag transfer functions.

Notes

To avoid small-signal stability issues, one take extreme care in setting the PI control gains `Kpvd`, `Kivd`, `Kpvq`, and `Kivq`, and the emulated inertia `M` and damping `D`.

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
bus		interface bus id			mandatory
gen		static generator index			mandatory
coi2		center of inertia 2 index			
Sn	S_n	Model MVA base	100	*MVA*	
fn	f	rated frequency	60		
Tc	T_c	switch time constant	0.010	*s*	
kw	k_ω	speed droop on active power (reciprocal of droop)	0	*p.u.*	non_negative,ipower
kv	k_v	reactive power droop on voltage	0	*p.u.*	non_negative,power
M	M	Emulated startup time constant (M=2H)	10	*s*	power
D	D	Emulated damping coefficient	0	*p.u.*	power
ra	r_a	resistance	0		z
xs	x_s	reactance	0.200		z
gammap	γ_P	P ratio of linked static gen	1		
gammaq	γ_Q	Q ratio of linked static gen	1		
Kpvd	kp_{vd}	vd controller proportional gain	0.500	*p.u.*	power
Kivd	ki_{vd}	vd controller integral gain	0.020	*p.u.*	power
Kpvq	kp_{vq}	vq controller proportional gain	0.500	*p.u.*	power
Kivq	ki_{vq}	vq controller integral gain	0.020	*p.u.*	power
Tiq	T_{Iq}		0.010		
Tid	T_{Id}		0.010		
ue	u_e	effective online status	0		

Variables

Name	Symbol	Type	Description	Unit	Properties
dw	dw	State	delta virtual rotor speed	*pu (Hz)*	v_str
delta	δ	State	virtual delta	*rad*	v_str
PIvd_xi	$PIvd_{\xi}$	State	Integrator output		v_str
PIvq_xi	$PIvq_{\xi}$	State	Integrator output		v_str
LGId_y	$LGId_y$	State	State in lag transfer function		v_str
LGIq_y	$LGIq_y$	State	State in lag transfer function		v_str
Pref2	$Pref_2$	Algeb	active power reference after adjusting by frequency		v_str
vref2	$vref_2$	Algeb	voltage reference after adjusted by reactive power		v_str
omega	ω	Algeb	virtual rotor speed	*pu (Hz)*	v_str
vd	vd	Algeb	d-axis voltage		v_str
vq	vq	Algeb	q-axis voltage		v_str
Pe	Pe	Algeb	active power injection from VSC		v_str
Qe	Qe	Algeb	reactive power injection from VSC		v_str
Id	Id	Algeb	d-axis current		v_str
Iq	Iq	Algeb	q-axis current		v_str
PIvd_y	$PIvd_y$	Algeb	PI output		v_str
PIvq_y	$PIvq_y$	Algeb	PI output		v_str
a	a	ExtAlgeb	Bus voltage angle		
v	v	ExtAlgeb	Bus voltage magnitude		
Idref	$Idref$	AliasAlgeb	Alias of PIvd_y		
Iqref	$Iqref$	AliasAlgeb	Alias of PIvq_y		

Initialization Equations

Name	Symbol	Type	Initial Value
dw	dw	State	0
delta	δ	State	a
PIvd_xi	$PIvd_{\xi}$	State	Id_0
PIvq_xi	$PIvq_{\xi}$	State	Iq_0
LGId_y	$LGId_y$	State	$PIvd_y$
LGIq_y	$LGIq_y$	State	$PIvq_y$
Pref2	$Pref_2$	Algeb	$Prefu$
vref2	$vref_2$	Algeb	$uvref$
omega	ω	Algeb	u
vd	vd	Algeb	vd_0
vq	vq	Algeb	vq_0
Pe	Pe	Algeb	$Pref$
Qe	Qe	Algeb	$Qref$
Id	Id	Algeb	Id_0
Iq	Iq	Algeb	Iq_0
PIvd_y	$PIvd_y$	Algeb	$Id_0 + Kpvd\,(vd - vref_2)$
PIvq_y	$PIvq_y$	Algeb	$Iq_0 + Kpvqvq$
a	a	ExtAlgeb	
v	v	ExtAlgeb	
Idref	$Idref$	AliasAlgeb	
Iqref	$Iqref$	AliasAlgeb	

Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
dw	dw	State	$-Ddw - Pe + Pref_2$	M
delta	δ	State	$2\pi dwfn$	
PIvd_xi	$PIvd_{\xi}$	State	$Kivd\,(vd - vref_2)$	
PIvq_xi	$PIvq_{\xi}$	State	$Kivqvq$	
LGId_y	$LGId_y$	State	$-LGId_y + PIvd_y$	T_{Id}
LGIq_y	$LGIq_y$	State	$-LGIq_y + PIvq_y$	T_{Iq}

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
Pref2	$Pref_2$	Algeb	$Prefue - Pref_2 - dwkw$
vref2	$vref_2$	Algeb	$kv\left(-Qe + Qrefue\right) + vref - vref_2$
omega	ω	Algeb	$dw - \omega + 1$
vd	vd	Algeb	$uev\cos\left(a - \delta\right) - vd$
vq	vq	Algeb	$uev\sin\left(a - \delta\right) - vq$
Pe	Pe	Algeb	$Idvd + Iqvq - Pe$
Qe	Qe	Algeb	$Idvq - Iqvd - Qe$
Id	Id	Algeb	$-Id + LGId_y$
Iq	Iq	Algeb	$-Iq + LGIq_y$
PIvd_y	$PIvd_y$	Algeb	$Kpvd\left(vd - vref_2\right) + PIvd_\xi - PIvd_y$
PIvq_y	$PIvq_y$	Algeb	$Kpvqvq + PIvq_\xi - PIvq_y$
a	a	ExtAlgeb	$-Peue$
v	v	ExtAlgeb	$-Qeue$
Idref	$Idref$	AliasAlgeb	0
Iqref	$Iqref$	AliasAlgeb	0

Services

Name	Symbol	Equation	Type
Pref	P_{ref}	$gammapp0s$	ConstService
Qref	Q_{ref}	$gammaqq0s$	ConstService
ixs	$1/xs$	$\frac{1}{xs}$	ConstService
Id0	I_{d0}	$\frac{Prefu}{v}$	ConstService
Iq0	I_{q0}	$-\frac{Qrefu}{v}$	ConstService
vd0	v_{d0}	uv	ConstService
vq0	v_{q0}	0	ConstService

Blocks

Name	Symbol	Type	Info
PIvd	$PIvd$	PIController	
PIvq	$PIvq$	PIController	
LGId	$LGId$	Lag	
LGIq	$LGIq$	Lag	

Setpoints

Name	Target	Info
pref	Pref	Initial P for the REGCV1 device
qref	Qref	Initial Q for the REGCV1 device
vref	vref	initial v of the static gen

Config Fields in [REGCV2]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

REGF1

Grid-forming inverter using droop.

Implementation of EPRI Memorandum

D. Ramasubramanian, "PROPOSAL FOR SUITE OF GENERIC GRID FORMING (GFM) POSITIVE SEQUENCE MODELS"

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
bus		interface bus id			mandatory
gen		static generator index			mandatory
Sn	S_n	Model MVA base	100	*MVA*	
rf	r_a	resistance	0		z
xf	x_s	reactance	0.200		z
Vdip	V_{dip}	V threshold to freeze states	0.800	*p.u.*	
Tfrz	T_{frz}	Time to keep state frozen	0		
PQFLAG		P/Q priority flag; 0-Q priority, 1-P priority	1	*bool*	
fn	f	rated frequency	60		
dwmax	$\Delta\omega_{max}$	maximum value of frequency deviation	75		
dwmin	$\Delta\omega_{min}$	minimum value of frequency deviation	-75		
wdrp	ω_{drp}	frequency droop percentage	0.033		

continues on next page

Table 28 – continued from previous page

Name	Symbol	Description	Default	Unit	Properties
Qdrp	Q_{drp}	Voltage droop percentage	0.045		
Tr	T_c	transducer time constant	0.005	s	
Te	T_e	ouput state time constant	0.005	s	
KPi	K_{Pi}	current control proportional gain	0.500		non_negative
KIi	K_{Ii}	current control integral gain	20		non_negative
KPv	K_{Pv}	voltage control proportional gain	3		non_negative
KIv	K_{Iv}	voltage control integral gain	10		non_negative
Pmax	P_{max}	max. active power	1		non_negative,power
Pmin	P_{min}	min. active power	-1		power
KPplim	K_{Pplim}	Kp for P limits	5		non_negative
KIplim	K_{Iplim}	KI for P limits	30		non_negative
Qmax	Q_{max}	max. reactive power	1		non_negative,power
Qmin	Q_{min}	min. reactive power	-1		power
KPqlim	K_{Pqlim}	Kp for Q limits	0.100		non_negative
KIqlim	K_{Iqlim}	KI for Q limits	1.500		non_negative
Tpm	T_{pm}	power signal input delay (3 Delta t)	0.025		
gammap	γ_P	P ratio of linked static gen	1		
gammaq	γ_Q	Q ratio of linked static gen	1		
ue	u_e	effective online status	0		

Variables

Name	Symbol	Type	Description	Unit	Properties
Psen_y	$Psen_y$	State	State in lag transfer function		v_str
Qsen_y	$Qsen_y$	State	State in lag transfer function		v_str
Psig_y	$Psig_y$	State	State in lag TF		v_str
Qsig_y	$Qsig_y$	State	State in lag TF		v_str
PIplim_xi	$PIplim_\xi$	State	Integrator output		v_str
PIqlim_xi	$PIqlim_\xi$	State	Integrator output		v_str
delta	δ	State	virtual delta	rad	v_str
PIvd_xi	$PIvd_\xi$	State	Integrator output		v_str
PIvq_xi	$PIvq_\xi$	State	Integrator output		v_str
PIId_xi	$PIId_\xi$	State	Integrator output		v_str
PIIq_xi	$PIIq_\xi$	State	Integrator output		v_str
udLag_y	$udLag_y$	State	State in lag transfer function		v_str
uqLag_y	$uqLag_y$	State	State in lag transfer function		v_str
ud	ud	AliasState	Alias of udLag_y		
uq	uq	AliasState	Alias of uqLag_y		
Paux	$Paux$	Algeb			v_str
Qaux	$Qaux$	Algeb			v_str
PIplim_y	$PIplim_y$	Algeb	PI output		v_str

continues on next page

Table 29 – continued from previous page

Name	Symbol	Type	Description	Unit	Properties
PIqlim_y	$PIqlim_y$	Algeb	PI output		v_str
vd	vd	Algeb	d-axis voltage		v_str
vq	vq	Algeb	q-axis voltage		v_str
Pe	Pe	Algeb	active power injection from VSC		v_str
Qe	Qe	Algeb	reactive power injection from VSC		v_str
Id	Id	Algeb	d-axis current		v_str
Iq	Iq	Algeb	q-axis current		v_str
dw_x	dw_x	Algeb	Value before limiter		v_str
vref2	$vref_2$	Algeb	voltage reference after droop		v_str
PIvd_y	$PIvd_y$	Algeb	PI output		v_str
PIvq_y	$PIvq_y$	Algeb	PI output		v_str
PIId_y	$PIId_y$	Algeb	PI output		v_str
PIIq_y	$PIIq_y$	Algeb	PI output		v_str
udref	$udref$	Algeb	ud reference		v_str
uqref	$uqref$	Algeb	uq reference		v_str
a	a	ExtAlgeb	Bus voltage angle		
v	v	ExtAlgeb	Bus voltage magnitude		
Idref	$Idref$	AliasAlgeb	Alias of PIvd_y		
Iqref	$Iqref$	AliasAlgeb	Alias of PIvq_y		

Initialization Equations

Name	Symbol	Type	Initial Value
Psen_y	$Psen_y$	State	Pe
Qsen_y	$Qsen_y$	State	Qe
Psig_y	$Psig_y$	State	$Paux + Psen_y$
Qsig_y	$Qsig_y$	State	$Qaux + Qsen_y$
PIplim_xi	$PIplim_\xi$	State	$Psen_y$
PIqlim_xi	$PIqlim_\xi$	State	$Qsen_y$
delta	δ	State	a
PIvd_xi	$PIvd_\xi$	State	Id_0
PIvq_xi	$PIvq_\xi$	State	Iq_0
PIId_xi	$PIId_\xi$	State	0.0
PIIq_xi	$PIIq_\xi$	State	0.0
udLag_y	$udLag_y$	State	$udref$
uqLag_y	$uqLag_y$	State	$uqref$
ud	ud	AliasState	
uq	uq	AliasState	
Paux	$Paux$	Algeb	$Paux_0$
Qaux	$Qaux$	Algeb	$Qaux_0$
PIplim_y	$PIplim_y$	Algeb	$KPplim\left(-Psen_y + Psig_y\right) + Psen_y$

continues on next page

Table 30 – continued from previous page

Name	Symbol	Type	Initial Value
PIqlim_y	$PIqlim_y$	Algeb	$KPqlim\left(-Qsen_y + Qsig_y\right) + Qsen_y$
vd	vd	Algeb	vd_0
vq	vq	Algeb	vq_0
Pe	Pe	Algeb	$Pref$
Qe	Qe	Algeb	$Qref$
Id	Id	Algeb	Id_0
Iq	Iq	Algeb	Iq_0
dw_x	dw_x	Algeb	$w_0 wdrp\left(PIplim_y - Psen_y\right)$
vref2	$vref_2$	Algeb	$uvref$
PIvd_y	$PIvd_y$	Algeb	$Id_0 + KPv\left(-vd + vref_2\right)$
PIvq_y	$PIvq_y$	Algeb	$Iq_0 - KPvvq$
PIId_y	$PIId_y$	Algeb	$KPi\left(-Id + PIvd_y\right)$
PIIq_y	$PIIq_y$	Algeb	$KPi\left(-Iq + PIvq_y\right)$
udref	$udref$	Algeb	$udref_0$
uqref	$uqref$	Algeb	$uqref_0$
a	a	ExtAlgeb	
v	v	ExtAlgeb	
Idref	$Idref$	AliasAlgeb	
Iqref	$Iqref$	AliasAlgeb	

Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
Psen_y	$Psen_y$	State	$Pe - Psen_y$	T_c
Qsen_y	$Qsen_y$	State	$Qe - Qsen_y$	T_c
Psig_y	$Psig_y$	State	$Paux + Psen_y - Psig_y$	T_{pm}
Qsig_y	$Qsig_y$	State	$Qaux + Qsen_y - Qsig_y$	T_{pm}
PIplim_xi	$PIplim_\xi$	State	$KIplim\left(-Psen_y + Psig_y\right)$	
PIqlim_xi	$PIqlim_\xi$	State	$KIqlim\left(-Qsen_y + Qsig_y\right)$	
delta	δ	State	$dw_{limzi} dw_x + dw_{limzl} dwmin + dw_{limzu} dwmax$	
PIvd_xi	$PIvd_\xi$	State	$KIv\left(-vd + vref_2\right)$	
PIvq_xi	$PIvq_\xi$	State	$-KIvvq$	
PIId_xi	$PIId_\xi$	State	$KIi\left(-Id + PIvd_y\right)$	
PIIq_xi	$PIIq_\xi$	State	$KIi\left(-Iq + PIvq_y\right)$	
udLag_y	$udLag_y$	State	$-udLag_y + udref$	T_e
uqLag_y	$uqLag_y$	State	$-uqLag_y + uqref$	T_e
ud	ud	AliasState	0	
uq	uq	AliasState	0	

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
Paux	$Paux$	Algeb	$-Paux + Paux_0$
Qaux	$Qaux$	Algeb	$-Qaux + Qaux_0$
PIplim_y	$PIplim_y$	Algeb	$KPplim\left(-Psen_y + Psig_y\right) + PIplim_\xi - PIplim_y$
PIqlim_y	$PIqlim_y$	Algeb	$KPqlim\left(-Qsen_y + Qsig_y\right) + PIqlim_\xi - PIqlim_y$
vd	vd	Algeb	$uev\cos\left(a - \delta\right) - vd$
vq	vq	Algeb	$uev\sin\left(a - \delta\right) - vq$
Pe	Pe	Algeb	$Idvd + Iqvq - Pe$
Qe	Qe	Algeb	$Idvq - Iqvd - Qe$
Id	Id	Algeb	$Idrf - Iqxf - udLag_y + vd$
Iq	Iq	Algeb	$Idxf + Iqrf - uqLag_y + vq$
dw_x	dw_x	Algeb	$-dw_x + w_0wdrp\left(PIplim_y - Psen_y\right)$
vref2	$vref_2$	Algeb	$Qdrp\left(PIqlim_yue - Qsen_y\right) + vref - vref_2$
PIvd_y	$PIvd_y$	Algeb	$KPv\left(-vd + vref_2\right) + PIvd_\xi - PIvd_y$
PIvq_y	$PIvq_y$	Algeb	$-KPvvq + PIvq_\xi - PIvq_y$
PIId_y	$PIId_y$	Algeb	$KPi\left(-Id + PIvd_y\right) + PIId_\xi - PIId_y$
PIIq_y	$PIIq_y$	Algeb	$KPi\left(-Iq + PIvq_y\right) + PIIq_\xi - PIIq_y$
udref	$udref$	Algeb	$Idrf - Iqxf + PIId_y - udref + vd$
uqref	$uqref$	Algeb	$Idxf + Iqrf + PIIq_y - uqref + vq$
a	a	ExtAlgeb	$-Peue$
v	v	ExtAlgeb	$-Qeue$
Idref	$Idref$	AliasAlgeb	0
Iqref	$Iqref$	AliasAlgeb	0

Services

Name	Symbol	Equation	Type
Pref	P_{ref}	$gammapp0s$	ConstService
Qref	Q_{ref}	$gammaqq0s$	ConstService
w0	$w0$	$2\pi fn$	ConstService
ixf	$1/xf$	$\frac{1}{xf}$	ConstService
Id0	I_{d0}	$\frac{Prefu}{v}$	ConstService
Iq0	I_{q0}	$-\frac{Qrefu}{v}$	ConstService
vd0	v_{d0}	uv	ConstService
vq0	v_{q0}	0	ConstService
Paux0	P_{aux0}	0	ConstService
Qaux0	Q_{aux0}	0	ConstService
udref0	u_{dref0}	$Id_0rf - Iq_0xf + vd_0$	ConstService
uqref0	u_{qref0}	$Id_0xf + Iq_0rf + vq_0$	ConstService

Discretes

Name	Symbol	Type	Info
Psig_lim	lim_{Psig}	AntiWindup	Limiter in Lag
Qsig_lim	lim_{Qsig}	AntiWindup	Limiter in Lag
dw_lim	lim_{dw}	HardLimiter	

Blocks

Name	Symbol	Type	Info
Psen	$Psen$	Lag	
Qsen	$Qsen$	Lag	
Psig	$Psig$	LagAntiWindup	
Qsig	$Qsig$	LagAntiWindup	
PIplim	$PIplim$	PIController	
PIqlim	$PIqlim$	PIController	
dw	dw	GainLimiter	
PIvd	$PIvd$	PIController	
PIvq	$PIvq$	PIController	
PIId	$PIId$	PIController	
PIIq	$PIIq$	PIController	
udLag	$udLag$	Lag	
uqLag	$uqLag$	Lag	

Setpoints

Name	Target	Info
pref	Pref	Initial P for the REGCV1 device
qref	Qref	Initial Q for the REGCV1 device
vref	vref	initial v of the static gen

Config Fields in [REGF1]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

REGF2

Grid-forming inverter with VSM control.

Implementation of EPRI Memorandum

D. Ramasubramanian, "PROPOSAL FOR SUITE OF GENERIC GRID FORMING (GFM) POSITIVE SEQUENCE MODELS"

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
bus		interface bus id			mandatory
gen		static generator index			mandatory
Sn	S_n	Model MVA base	100	*MVA*	
rf	r_a	resistance	0		z
xf	x_s	reactance	0.200		z
Vdip	V_{dip}	V threshold to freeze states	0.800	*p.u.*	
Tfrz	T_{frz}	Time to keep state frozen	0		
PQFLAG		P/Q priority flag; 0-Q priority, 1-P priority	1	*bool*	
fn	f	rated frequency	60		
dwmax	$\Delta\omega_{max}$	maximum value of frequency deviation	75		
dwmin	$\Delta\omega_{min}$	minimum value of frequency deviation	-75		
wdrp	ω_{drp}	frequency droop percentage	0.033		
Qdrp	Q_{drp}	Voltage droop percentage	0.045		
Tr	T_c	transducer time constant	0.005	*s*	
Te	T_e	ouput state time constant	0.005	*s*	
KPi	K_{Pi}	current control proportional gain	0.500		non_negative
KIi	K_{Ii}	current control integral gain	20		non_negative
KPv	K_{Pv}	voltage control proportional gain	3		non_negative
KIv	K_{Iv}	voltage control integral gain	10		non_negative
Pmax	P_{max}	max. active power	1		non_negative,power
Pmin	P_{min}	min. active power	-1		power
KPplim	K_{Pplim}	Kp for P limits	5		non_negative
KIplim	K_{Iplim}	KI for P limits	30		non_negative
Qmax	Q_{max}	max. reactive power	1		non_negative,power
Qmin	Q_{min}	min. reactive power	-1		power
KPqlim	K_{Pqlim}	Kp for Q limits	0.100		non_negative
KIqlim	K_{Iqlim}	KI for Q limits	1.500		non_negative
Tpm	T_{pm}	power signal input delay (3 Delta t)	0.025		
gammap	γ_P	P ratio of linked static gen	1		
gammaq	γ_Q	Q ratio of linked static gen	1		

continues on next page

Table 31 – continued from previous page

Name	Symbol	Description	Default	Unit	Properties
mf	M_f	VSM inertia constant	0.150		
dd	d_d	VSM damping factor	0.110		
pll		PLL device idx (optional)			
ue	u_e	effective online status	0		

Variables

Name	Symbol	Type	Description	Unit	Properties
Psen_y	$Psen_y$	State	State in lag transfer function		v_str
Qsen_y	$Qsen_y$	State	State in lag transfer function		v_str
Psig_y	$Psig_y$	State	State in lag TF		v_str
Qsig_y	$Qsig_y$	State	State in lag TF		v_str
PIplim_xi	$PIplim_\xi$	State	Integrator output		v_str
PIqlim_xi	$PIqlim_\xi$	State	Integrator output		v_str
delta	δ	State	virtual delta	*rad*	v_str
INTw_y	$INTw_y$	State	Integrator output		v_str
PIvd_xi	$PIvd_\xi$	State	Integrator output		v_str
PIvq_xi	$PIvq_\xi$	State	Integrator output		v_str
PIId_xi	$PIId_\xi$	State	Integrator output		v_str
PIIq_xi	$PIIq_\xi$	State	Integrator output		v_str
udLag_y	$udLag_y$	State	State in lag transfer function		v_str
uqLag_y	$uqLag_y$	State	State in lag transfer function		v_str
ud	ud	AliasState	Alias of udLag_y		
uq	uq	AliasState	Alias of uqLag_y		
Paux	$Paux$	Algeb			v_str
Qaux	$Qaux$	Algeb			v_str
PIplim_y	$PIplim_y$	Algeb	PI output		v_str
PIqlim_y	$PIqlim_y$	Algeb	PI output		v_str
vd	vd	Algeb	d-axis voltage		v_str
vq	vq	Algeb	q-axis voltage		v_str
Pe	Pe	Algeb	active power injection from VSC		v_str
Qe	Qe	Algeb	reactive power injection from VSC		v_str
Id	Id	Algeb	d-axis current		v_str
Iq	Iq	Algeb	q-axis current		v_str
wref	$wref$	Algeb	speed ref	*pu*	v_str
dw_x	dw_x	Algeb	Value before limiter		v_str
vref2	$vref_2$	Algeb	voltage reference after droop		v_str
PIvd_y	$PIvd_y$	Algeb	PI output		v_str
PIvq_y	$PIvq_y$	Algeb	PI output		v_str
PIId_y	$PIId_y$	Algeb	PI output		v_str
PIIq_y	$PIIq_y$	Algeb	PI output		v_str

continues on next page

Table 32 – continued from previous page

Name	Symbol	Type	Description	Unit	Properties
udref	$udref$	Algeb	ud reference		v_str
uqref	$uqref$	Algeb	uq reference		v_str
a	a	ExtAlgeb	Bus voltage angle		
v	v	ExtAlgeb	Bus voltage magnitude		
plldw	$plldw$	ExtAlgeb	PLL measured freq. deviation		
Idref	$Idref$	AliasAlgeb	Alias of PIvd_y		
Iqref	$Iqref$	AliasAlgeb	Alias of PIvq_y		

Initialization Equations

Name	Symbol	Type	Initial Value
Psen_y	$Psen_y$	State	Pe
Qsen_y	$Qsen_y$	State	Qe
Psig_y	$Psig_y$	State	$Paux + Psen_y$
Qsig_y	$Qsig_y$	State	$Qaux + Qsen_y$
PIplim_xi	$PIplim_\xi$	State	$Psen_y$
PIqlim_xi	$PIqlim_\xi$	State	$Qsen_y$
delta	δ	State	a
INTw_y	$INTw_y$	State	$wref$
PIvd_xi	$PIvd_\xi$	State	Id_0
PIvq_xi	$PIvq_\xi$	State	Iq_0
PIId_xi	$PIId_\xi$	State	0.0
PIIq_xi	$PIIq_\xi$	State	0.0
udLag_y	$udLag_y$	State	$udref$
uqLag_y	$uqLag_y$	State	$uqref$
ud	ud	AliasState	
uq	uq	AliasState	
Paux	$Paux$	Algeb	$Paux_0$
Qaux	$Qaux$	Algeb	$Qaux_0$
PIplim_y	$PIplim_y$	Algeb	$KPplim\left(-Psen_y + Psig_y\right) + Psen_y$
PIqlim_y	$PIqlim_y$	Algeb	$KPqlim\left(-Qsen_y + Qsig_y\right) + Qsen_y$
vd	vd	Algeb	vd_0
vq	vq	Algeb	vq_0
Pe	Pe	Algeb	$Pref$
Qe	Qe	Algeb	$Qref$
Id	Id	Algeb	Id_0
Iq	Iq	Algeb	Iq_0
wref	$wref$	Algeb	1
dw_x	dw_x	Algeb	$w_0\left(INTw_y - wref\right)$
vref2	$vref_2$	Algeb	$uvref$
PIvd_y	$PIvd_y$	Algeb	$Id_0 + KPv\left(-vd + vref_2\right)$

continues on next page

Table 33 – continued from previous page

Name	Symbol	Type	Initial Value
PIvq_y	$PIvq_y$	Algeb	$Iq_0 - KPvvq$
PIId_y	$PIId_y$	Algeb	$KPi\left(-Id + PIvd_y\right)$
PIIq_y	$PIIq_y$	Algeb	$KPi\left(-Iq + PIvq_y\right)$
udref	$udref$	Algeb	$udref_0$
uqref	$uqref$	Algeb	$uqref_0$
a	a	ExtAlgeb	
v	v	ExtAlgeb	
plldw	$plldw$	ExtAlgeb	
Idref	$Idref$	AliasAlgeb	
Iqref	$Iqref$	AliasAlgeb	

Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
Psen_y	$Psen_y$	State	$Pe - Psen_y$	T_c
Qsen_y	$Qsen_y$	State	$Qe - Qsen_y$	T_c
Psig_y	$Psig_y$	State	$Paux + Psen_y - Psig_y$	T_{pm}
Qsig_y	$Qsig_y$	State	$Qaux + Qsen_y - Qsig_y$	T_{pm}
PI-plim_xi	$PIplim_\xi$	State	$KIplim\left(-Psen_y + Psig_y\right)$	
PIqlim_xi	$PIqlim_\xi$	State	$KIqlim\left(-Qsen_y + Qsig_y\right)$	
delta	δ	State	$dw_{limzi}dw_x + dw_{limzl}dwmin + dw_{limzu}dwmax$	
INTw_y	$INTw_y$	State	$-INTw_y + ddplldwwdrp + wdrp\left(PIplim_y - Psen_y\right) + wref$	T_{int}
PIvd_xi	$PIvd_\xi$	State	$KIv\left(-vd + vref_2\right)$	
PIvq_xi	$PIvq_\xi$	State	$-KIvvq$	
PIId_xi	$PIId_\xi$	State	$KIi\left(-Id + PIvd_y\right)$	
PIIq_xi	$PIIq_\xi$	State	$KIi\left(-Iq + PIvq_y\right)$	
udLag_y	$udLag_y$	State	$-udLag_y + udref$	T_e
uqLag_y	$uqLag_y$	State	$-uqLag_y + uqref$	T_e
ud	ud	AliasState	0	
uq	uq	AliasState	0	

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
Paux	$Paux$	Algeb	$-Paux + Paux_0$
Qaux	$Qaux$	Algeb	$-Qaux + Qaux_0$
PIplim_y	$PIplim_y$	Algeb	$KPplim\left(-Psen_y + Psig_y\right) + PIplim_\xi - PIplim_y$
PIqlim_y	$PIqlim_y$	Algeb	$KPqlim\left(-Qsen_y + Qsig_y\right) + PIqlim_\xi - PIqlim_y$
vd	vd	Algeb	$uev\cos\left(a - \delta\right) - vd$
vq	vq	Algeb	$uev\sin\left(a - \delta\right) - vq$
Pe	Pe	Algeb	$Idvd + Iqvq - Pe$
Qe	Qe	Algeb	$Idvq - Iqvd - Qe$
Id	Id	Algeb	$Idrf - Iqxf - udLag_y + vd$
Iq	Iq	Algeb	$Idxf + Iqrf - uqLag_y + vq$
wref	$wref$	Algeb	$1 - wref$
dw_x	dw_x	Algeb	$-dw_x + w_0\left(INTw_y - wref\right)$
vref2	$vref_2$	Algeb	$Qdrp\left(PIqlim_y ue - Qsen_y\right) + vref - vref_2$
PIvd_y	$PIvd_y$	Algeb	$KPv\left(-vd + vref_2\right) + PIvd_\xi - PIvd_y$
PIvq_y	$PIvq_y$	Algeb	$-KPvvq + PIvq_\xi - PIvq_y$
PIId_y	$PIId_y$	Algeb	$KPi\left(-Id + PIvd_y\right) + PIId_\xi - PIId_y$
PIIq_y	$PIIq_y$	Algeb	$KPi\left(-Iq + PIvq_y\right) + PIIq_\xi - PIIq_y$
udref	$udref$	Algeb	$Idrf - Iqxf + PIId_y - udref + vd$
uqref	$uqref$	Algeb	$Idxf + Iqrf + PIIq_y - uqref + vq$
a	a	ExtAlgeb	$-Peue$
v	v	ExtAlgeb	$-Qeue$
plldw	$plldw$	ExtAlgeb	0
Idref	$Idref$	AliasAlgeb	0
Iqref	$Iqref$	AliasAlgeb	0

Services

Name	Symbol	Equation	Type
Pref	P_{ref}	$gammapp0s$	ConstService
Qref	Q_{ref}	$gammaqq0s$	ConstService
w0	$w0$	$2\pi fn$	ConstService
ixf	$1/xf$	$\frac{1}{xf}$	ConstService
Id0	I_{d0}	$\frac{Prefu}{v}$	ConstService
Iq0	I_{q0}	$-\frac{Qrefu}{v}$	ConstService
vd0	v_{d0}	uv	ConstService
vq0	v_{q0}	0	ConstService
Paux0	P_{aux0}	0	ConstService
Qaux0	Q_{aux0}	0	ConstService
Tint	T_{int}	$mfwdrp$	ConstService
udref0	u_{dref0}	$Id_0rf - Iq_0xf + vd_0$	ConstService
uqref0	u_{qref0}	$Id_0xf + Iq_0rf + vq_0$	ConstService

Discretes

Name	Symbol	Type	Info
Psig_lim	lim_{Psig}	AntiWindup	Limiter in Lag
Qsig_lim	lim_{Qsig}	AntiWindup	Limiter in Lag
dw_lim	lim_{dw}	HardLimiter	

Blocks

Name	Symbol	Type	Info
Psen	$Psen$	Lag	
Qsen	$Qsen$	Lag	
Psig	$Psig$	LagAntiWindup	
Qsig	$Qsig$	LagAntiWindup	
PIplim	$PIplim$	PIController	
PIqlim	$PIqlim$	PIController	
INTw	$INTw$	Integrator	
dw	dw	GainLimiter	
PIvd	$PIvd$	PIController	
PIvq	$PIvq$	PIController	
PIId	$PIId$	PIController	
PIIq	$PIIq$	PIController	
udLag	$udLag$	Lag	
uqLag	$uqLag$	Lag	

Setpoints

Name	Target	Info
pref	Pref	Initial P for the REGCV1 device
qref	Qref	Initial Q for the REGCV1 device
vref	vref	initial v of the static gen

Config Fields in [REGF2]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

REGF3

Grid-forming inverter with dVOC.

Implementation of EPRI Memorandum

D. Ramasubramanian, "PROPOSAL FOR SUITE OF GENERIC GRID FORMING (GFM) POSITIVE SEQUENCE MODELS"

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
bus		interface bus id			mandatory
gen		static generator index			mandatory
Sn	S_n	Model MVA base	100	*MVA*	
rf	r_a	resistance	0		z
xf	x_s	reactance	0.200		z
Vdip	V_{dip}	V threshold to freeze states	0.800	*p.u.*	
Tfrz	T_{frz}	Time to keep state frozen	0		
PQFLAG		P/Q priority flag; 0-Q priority, 1-P priority	1	*bool*	
fn	f	rated frequency	60		
dwmax	$\Delta\omega_{max}$	maximum value of frequency deviation	75		
dwmin	$\Delta\omega_{min}$	minimum value of frequency deviation	-75		
wdrp	ω_{drp}	frequency droop percentage	0.033		
Qdrp	Q_{drp}	Voltage droop percentage	0.045		
Tr	T_c	transducer time constant	0.005	*s*	
Te	T_e	ouput state time constant	0.005	*s*	
KPi	K_{Pi}	current control proportional gain	0.500		non_negative
KIi	K_{Ii}	current control integral gain	20		non_negative
KPv	K_{Pv}	voltage control proportional gain	3		non_negative
KIv	K_{Iv}	voltage control integral gain	10		non_negative
Pmax	P_{max}	max. active power	1		non_negative,power
Pmin	P_{min}	min. active power	-1		power
KPplim	K_{Pplim}	Kp for P limits	5		non_negative
KIplim	K_{Iplim}	KI for P limits	30		non_negative
Qmax	Q_{max}	max. reactive power	1		non_negative,power
Qmin	Q_{min}	min. reactive power	-1		power
KPqlim	K_{Pqlim}	Kp for Q limits	0.100		non_negative
KIqlim	K_{Iqlim}	KI for Q limits	1.500		non_negative
Tpm	T_{pm}	power signal input delay (3 Delta t)	0.025		
gammap	γ_P	P ratio of linked static gen	1		
gammaq	γ_Q	Q ratio of linked static gen	1		
ue	u_e	effective online status	0		

Variables

Name	Symbol	Type	Description	Unit	Properties
Psen_y	$Psen_y$	State	State in lag transfer function		v_str
Qsen_y	$Qsen_y$	State	State in lag transfer function		v_str
Psig_y	$Psig_y$	State	State in lag TF		v_str
Qsig_y	$Qsig_y$	State	State in lag TF		v_str
PIplim_xi	$PIplim_\xi$	State	Integrator output		v_str
PIqlim_xi	$PIqlim_\xi$	State	Integrator output		v_str
delta	δ	State	virtual delta	*rad*	v_str
vref2	$vref_2$	State			v_str
PIvd_xi	$PIvd_\xi$	State	Integrator output		v_str
PIvq_xi	$PIvq_\xi$	State	Integrator output		v_str
PIId_xi	$PIId_\xi$	State	Integrator output		v_str
PIIq_xi	$PIIq_\xi$	State	Integrator output		v_str
udLag_y	$udLag_y$	State	State in lag transfer function		v_str
uqLag_y	$uqLag_y$	State	State in lag transfer function		v_str
ud	ud	AliasState	Alias of udLag_y		
uq	uq	AliasState	Alias of uqLag_y		
Paux	$Paux$	Algeb			v_str
Qaux	$Qaux$	Algeb			v_str
PIplim_y	$PIplim_y$	Algeb	PI output		v_str
PIqlim_y	$PIqlim_y$	Algeb	PI output		v_str
vd	vd	Algeb	d-axis voltage		v_str
vq	vq	Algeb	q-axis voltage		v_str
Pe	Pe	Algeb	active power injection from VSC		v_str
Qe	Qe	Algeb	reactive power injection from VSC		v_str
Id	Id	Algeb	d-axis current		v_str
Iq	Iq	Algeb	q-axis current		v_str
wref	$wref$	Algeb	speed ref	*pu*	v_str
Vref	$Vref$	Algeb			v_str
dw_x	dw_x	Algeb	Value before limiter		v_str
derv	$derv$	Algeb	input to voltage integrator		v_str
PIvd_y	$PIvd_y$	Algeb	PI output		v_str
PIvq_y	$PIvq_y$	Algeb	PI output		v_str
PIId_y	$PIId_y$	Algeb	PI output		v_str
PIIq_y	$PIIq_y$	Algeb	PI output		v_str
udref	$udref$	Algeb	ud reference		v_str
uqref	$uqref$	Algeb	uq reference		v_str
a	a	ExtAlgeb	Bus voltage angle		
v	v	ExtAlgeb	Bus voltage magnitude		
Idref	$Idref$	AliasAlgeb	Alias of PIvd_y		
Iqref	$Iqref$	AliasAlgeb	Alias of PIvq_y		

Initialization Equations

Name	Symbol	Type	Initial Value
Psen_y	$Psen_y$	State	Pe
Qsen_y	$Qsen_y$	State	Qe
Psig_y	$Psig_y$	State	$Paux + Psen_y$
Qsig_y	$Qsig_y$	State	$Qaux + Qsen_y$
PIplim_xi	$PIplim_\xi$	State	$Psen_y$
PIqlim_xi	$PIqlim_\xi$	State	$Qsen_y$
delta	δ	State	a
vref2	$vref_2$	State	vd
PIvd_xi	$PIvd_\xi$	State	Id_0
PIvq_xi	$PIvq_\xi$	State	Iq_0
PIId_xi	$PIId_\xi$	State	0.0
PIIq_xi	$PIIq_\xi$	State	0.0
udLag_y	$udLag_y$	State	$udref$
uqLag_y	$uqLag_y$	State	$uqref$
ud	ud	AliasState	
uq	uq	AliasState	
Paux	$Paux$	Algeb	$Paux_0$
Qaux	$Qaux$	Algeb	$Qaux_0$
PIplim_y	$PIplim_y$	Algeb	$KPplim\left(-Psen_y + Psig_y\right) + Psen_y$
PIqlim_y	$PIqlim_y$	Algeb	$KPqlim\left(-Qsen_y + Qsig_y\right) + Qsen_y$
vd	vd	Algeb	vd_0
vq	vq	Algeb	vq_0
Pe	Pe	Algeb	$Pref$
Qe	Qe	Algeb	$Qref$
Id	Id	Algeb	Id_0
Iq	Iq	Algeb	Iq_0
wref	$wref$	Algeb	1
Vref	$Vref$	Algeb	$vref$
dw_x	dw_x	Algeb	$\frac{w_0 wdrp(PIplim_y - Psen_y)}{vref_2^2}$
derv	$derv$	Algeb	0
PIvd_y	$PIvd_y$	Algeb	$Id_0 + KPv\left(-vd + vref_2\right)$
PIvq_y	$PIvq_y$	Algeb	$Iq_0 - KPvvq$
PIId_y	$PIId_y$	Algeb	$KPi\left(-Id + PIvd_y\right)$
PIIq_y	$PIIq_y$	Algeb	$KPi\left(-Iq + PIvq_y\right)$
udref	$udref$	Algeb	$udref_0$
uqref	$uqref$	Algeb	$uqref_0$
a	a	ExtAlgeb	
v	v	ExtAlgeb	
Idref	$Idref$	AliasAlgeb	
Iqref	$Iqref$	AliasAlgeb	

Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
Psen_y	$Psen_y$	State	$Pe - Psen_y$	T_c
Qsen_y	$Qsen_y$	State	$Qe - Qsen_y$	T_c
Psig_y	$Psig_y$	State	$Paux + Psen_y - Psig_y$	T_{pm}
Qsig_y	$Qsig_y$	State	$Qaux + Qsen_y - Qsig_y$	T_{pm}
PIplim_xi	$PIplim_\xi$	State	$KIplim\left(-Psen_y + Psig_y\right)$	
PIqlim_xi	$PIqlim_\xi$	State	$KIqlim\left(-Qsen_y + Qsig_y\right)$	
delta	δ	State	$dw_{limzi}dw_x + dw_{limzl}dwmin + dw_{limzu}dwmax$	
vref2	$vref_2$	State	$dervw_0$	
PIvd_xi	$PIvd_\xi$	State	$KIv\left(-vd + vref_2\right)$	
PIvq_xi	$PIvq_\xi$	State	$-KIvvq$	
PIId_xi	$PIId_\xi$	State	$KIi\left(-Id + PIvd_y\right)$	
PIIq_xi	$PIIq_\xi$	State	$KIi\left(-Iq + PIvq_y\right)$	
udLag_y	$udLag_y$	State	$-udLag_y + udref$	T_e
uqLag_y	$uqLag_y$	State	$-uqLag_y + uqref$	T_e
ud	ud	AliasState	0	
uq	uq	AliasState	0	

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
Paux	$Paux$	Algeb	$-Paux + Paux_0$
Qaux	$Qaux$	Algeb	$-Qaux + Qaux_0$
PI-plim_y	$PIplim_y$	Algeb	$KPplim\left(-Psen_y + Psig_y\right) + PIplim_\xi - PIplim_y$
PIqlim_y	$PIqlim_y$	Algeb	$KPqlim\left(-Qsen_y + Qsig_y\right) + PIqlim_\xi - PIqlim_y$
vd	vd	Algeb	$uev\cos\left(a - \delta\right) - vd$
vq	vq	Algeb	$uev\sin\left(a - \delta\right) - vq$
Pe	Pe	Algeb	$Idvd + Iqvq - Pe$
Qe	Qe	Algeb	$Idvq - Iqvd - Qe$
Id	Id	Algeb	$Idrf - Iqxf - udLag_y + vd$
Iq	Iq	Algeb	$Idxf + Iqrf - uqLag_y + vq$
wref	$wref$	Algeb	$1 - wref$
Vref	$Vref$	Algeb	$-Vref + vref$
dw_x	dw_x	Algeb	$-dw_x + \frac{w_0 wdrp(PIplim_y - Psen_y)}{vref_2^2}$
derv	$derv$	Algeb	$Kdvocvref_2\left(Vref - vref_2\right)\left(Vref + vref_2\right) - derv + \frac{wdrp(PIqlim_y - Qsen_y)}{vref_2}$
PIvd_y	$PIvd_y$	Algeb	$KPv\left(-vd + vref_2\right) + PIvd_\xi - PIvd_y$
PIvq_y	$PIvq_y$	Algeb	$-KPvvq + PIvq_\xi - PIvq_y$
PIId_y	$PIId_y$	Algeb	$KPi\left(-Id + PIvd_y\right) + PIId_\xi - PIId_y$
PIIq_y	$PIIq_y$	Algeb	$KPi\left(-Iq + PIvq_y\right) + PIIq_\xi - PIIq_y$
udref	$udref$	Algeb	$Idrf - Iqxf + PIId_y - udref + vd$
uqref	$uqref$	Algeb	$Idxf + Iqrf + PIIq_y - uqref + vq$
a	a	ExtAlgeb	$-Peue$
v	v	ExtAlgeb	$-Qeue$
Idref	$Idref$	AliasAlgeb	0
Iqref	$Iqref$	AliasAlgeb	0

Services

Name	Symbol	Equation	Type
Pref	P_{ref}	$gammapp0s$	ConstService
Qref	Q_{ref}	$gammaqq0s$	ConstService
w0	$w0$	$2\pi fn$	ConstService
ixf	$1/xf$	$\frac{1}{xf}$	ConstService
Id0	I_{d0}	$\frac{Prefu}{v}$	ConstService
Iq0	I_{q0}	$-\frac{Qrefu}{v}$	ConstService
vd0	v_{d0}	uv	ConstService
vq0	v_{q0}	0	ConstService
Paux0	P_{aux0}	0	ConstService
Qaux0	Q_{aux0}	0	ConstService
Kdvoc	$Kdvoc$	$\frac{400000000wdrp}{\left(99990000-2(100-100Qdrp)^2\right)^2}$	ConstService
udref0	u_{dref0}	$Id_0rf - Iq_0xf + vd_0$	ConstService
uqref0	u_{qref0}	$Id_0xf + Iq_0rf + vq_0$	ConstService

Discretes

Name	Symbol	Type	Info
Psig_lim	lim_{Psig}	AntiWindup	Limiter in Lag
Qsig_lim	lim_{Qsig}	AntiWindup	Limiter in Lag
dw_lim	lim_{dw}	HardLimiter	

Blocks

Name	Symbol	Type	Info
Psen	$Psen$	Lag	
Qsen	$Qsen$	Lag	
Psig	$Psig$	LagAntiWindup	
Qsig	$Qsig$	LagAntiWindup	
PIplim	$PIplim$	PIController	
PIqlim	$PIqlim$	PIController	
dw	dw	GainLimiter	
PIvd	$PIvd$	PIController	
PIvq	$PIvq$	PIController	
PIId	$PIId$	PIController	
PIIq	$PIIq$	PIController	
udLag	$udLag$	Lag	
uqLag	$uqLag$	Lag	

Setpoints

Name	Target	Info
pref	Pref	Initial P for the REGCV1 device
qref	Qref	Initial Q for the REGCV1 device
vref	vref	initial v of the static gen

Config Fields in [REGF3]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

5.3.26 RenGovernor

Renewable turbine governor group.

Common Parameters: u, name, ree, w0, Sn, Pe0

Common Variables: Pm, wr0, wt, wg, s3_y

Available models: *WTDTA1*, *WTDS*

WTDTA1

WTDTA wind turbine drive-train model.

One can set `Htfrac` to `0` to simulate a single-mass drive train. `Htfrac` has to be within `[0, 1]`

User-provided reference speed should be specified in parameter *w0*. Internally, *w0* is set to the algebraic variable *wr0*.

Note for PSS/E dyr parser:

In PSS/E doc, *Freq1* is said to be Hz, but exported data from PSS/E 34 uses per unit. ANDES requires `Freq1` in per unit frequency.

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
ree		Renewable exciter idx			mandatory
H	H_t	Total inertia constant	3	*MWs/MVA*	non_zero,non_negative,power
DAMP	$Damp$	Damp coefficient	0	*p.u. (gen base)*	power
Htfrac	H_{tfrac}	Turbine inertia fraction (Hturb/H)	0.500		
Freq1	$Freq1$	First shaft torsional resonant frequency, p.u. (Hz)	1	*p.u. (Hz)*	
Dshaft	D_{shaft}	Shaft damping factor	1	*p.u. (gen base)*	power
w0	ω_0	Default speed if not using a torque model	1	*p.u.*	
ue	u_e	effective online status	0		
reg			0		
Sn	S_n		0		

Variables

Name	Symbol	Type	Description	Unit	Properties
s1_y	s_{1y}	State	Integrator output		v_str
s2_y	s_{2y}	State	Integrator output		v_str
s3_y	s_{3y}	State	Integrator output		v_str
wt	wt	AliasState	Alias of s1_y		
wg	wg	AliasState	Alias of s2_y		
wr0	wr_0	Algeb	speed set point	*p.u.*	v_str
Pm	Pm	Algeb	Mechanical power		v_str
pd	pd	Algeb	Output after damping		v_str
wge	wge	ExtAlgeb			
Pe	Pe	ExtAlgeb	Retrieved Pe of RenGen		

Initialization Equations

Name	Symbol	Type	Initial Value
s1_y	s_{1y}	State	wr_0
s2_y	s_{2y}	State	wr_0
s3_y	s_{3y}	State	$\frac{Pe_0}{wr_0}$
wt	wt	AliasState	
wg	wg	AliasState	
wr0	wr_0	Algeb	w_0
Pm	Pm	Algeb	Pe_0
pd	pd	Algeb	0
wge	wge	ExtAlgeb	
Pe	Pe	ExtAlgeb	

Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
s1_y	s_{1y}	State	$\frac{1.0Pm}{s_{1y}} - 1.0pd - 1.0s_{3y}$	$2H_t$
s2_y	s_{2y}	State	$-1.0DAMP\left(s_{2y} - w_0\right) - \frac{1.0Pe}{s_{2y}} + 1.0pd + 1.0s_{3y}$	$2H_g$
s3_y	s_{3y}	State	$Kshaft\left(s_{1y} - s_{2y}\right)$	1.0
wt	wt	AliasState	0	
wg	wg	AliasState	0	

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
wr0	wr_0	Algeb	$w_0 - wr_0$
Pm	Pm	Algeb	$Pe_0 - Pm$
pd	pd	Algeb	$Dshaft\left(s_{1y} - s_{2y}\right) - pd$
wge	wge	ExtAlgeb	$ue\left(s_{2y} - 1.0\right)$
Pe	Pe	ExtAlgeb	0

Services

Name	Symbol	Equation	Type
Ht2	$2H_t$	$2HHtfrac$	ConstService
Hg2	$2H_g$	$2H\left(1 - Htfrac\right)$	ConstService
Kshaft	K_{shaft}	$\frac{0.5Freq_1^2Hg_2Ht_2}{H}$	ConstService

Blocks

Name	Symbol	Type	Info
s1	$s1$	Integrator	
s2	$s2$	Integrator	
s3	$s3$	Integrator	

Config Fields in [WTDTA1]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

WTDS

Custom wind turbine model with a single swing-equation.

This model is used to simulate the mechanical swing of the combined machine and turbine mass. The speed output is `s1_y` which will be fed to `RenExciter.wg`.

`PFLAG` needs to be set to `1` in exciter to consider speed for Pref.

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
ree		Renewable exciter idx			mandatory
H	H_t	Total inertia	3	*MWs/MVA*	non_zero,non_negative,power
D	D_{shaft}	Damping coefficient	1	*p.u.*	power
w0	ω_0	Default speed if not using a torque model	1	*p.u.*	
ue	u_e	effective online status	0		
reg			0		
Sn	S_n		0		

Variables

Name	Symbol	Type	Description	Unit	Properties
s1_y	s_{1y}	State	Integrator output		v_str
s3_y	s_{3y}	State	Unused state variable		
wt	wt	AliasState	Alias of s1_y		
wg	wg	AliasState	Alias of s1_y		
Pm	Pm	Algeb	Mechanical power		v_str
wr0	wr_0	Algeb	speed set point	*p.u.*	v_str
wge	wge	ExtAlgeb			
Pe	Pe	ExtAlgeb	Retrieved Pe of RenGen		

Initialization Equations

Name	Symbol	Type	Initial Value
s1_y	s_{1y}	State	wr_0
s3_y	s_{3y}	State	
wt	wt	AliasState	
wg	wg	AliasState	
Pm	Pm	Algeb	Pe_0
wr0	wr_0	Algeb	w_0
wge	wge	ExtAlgeb	
Pe	Pe	ExtAlgeb	

Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
s1_y	s_{1y}	State	$-1.0D\left(s_{1y} - wr_0\right) + \frac{1.0(-Pe+Pm)}{wge}$	$2H$
s3_y	s_{3y}	State	0	
wt	wt	AliasState	0	
wg	wg	AliasState	0	

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
Pm	Pm	Algeb	$Pe_0 - Pm$
wr0	wr_0	Algeb	$w_0 - wr_0$
wge	wge	ExtAlgeb	$ue\left(s_{1y} - 1.0\right)$
Pe	Pe	ExtAlgeb	0

Services

Name	Symbol	Equation	Type
H2	$2H$	$2H$	ConstService
Kshaft	K_{shaft}	1.0	ConstService

Blocks

Name	Symbol	Type	Info
s1	$s1$	Integrator	

Config Fields in [WTDS]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

5.3.27 RenPitch

Renewable generator pitch controller group.

Common Parameters: u, name, rea

Available models: *WTPTA1*

WTPTA1

Wind turbine pitch control model.

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
rea		Renewable aerodynamics model idx			mandatory
Kiw	K_{iw}	Pitch-control integral gain	0.100	*p.u.*	
Kpw	K_{pw}	Pitch-control proportional gain	0	*p.u.*	
Kic	K_{ic}	Pitch-compensation integral gain	0.100	*p.u.*	
Kpc	K_{pc}	Pitch-compensation proportional gain	0	*p.u.*	
Kcc	K_{cc}	Gain for P diff	0	*p.u.*	
Tp	T_θ	Blade response time const.	0.300	*s*	
thmax	θ_{max}	Max. pitch angle	30	*deg.*	
thmin	θ_{min}	Min. pitch angle	0	*deg.*	
dthmax	θ_{max}	Max. pitch angle rate	5	*deg.*	
dthmin	θ_{min}	Min. pitch angle rate	-5	*deg.*	
ue	u_e	effective online status	0		
rego			0		
ree			0		

Variables

Name	Symbol	Type	Description	Unit	Properties
PIc_xi	PIc_ξ	State	Integrator output		v_str
PIw_xi	PIw_ξ	State	Integrator output		v_str
LG_y	LG_y	State	State in lag TF		v_str
wt	wt	ExtState			
Pord	$Pord$	ExtState			
PIc_yul	PIc_{yul}	Algeb			v_str
PIc_y	PIc_y	Algeb	PI output		v_str
wref	$wref$	Algeb	optional speed reference		v_str
PIw_yul	PIw_{yul}	Algeb			v_str
PIw_y	PIw_y	Algeb	PI output		v_str
theta	θ	ExtAlgeb			
Pref	$Pref$	ExtAlgeb			

Initialization Equations

Name	Symbol	Type	Initial Value
PIc_xi	PIc_{ξ}	State	0.0
PIw_xi	PIw_{ξ}	State	0.0
LG_y	LG_y	State	$1.0PIc_y + 1.0PIw_y$
wt	wt	ExtState	
Pord	$Pord$	ExtState	
PIc_yul	PIc_{yul}	Algeb	$Kpc\left(Pord - Pref\right)$
PIc_y	PIc_y	Algeb	$PIc_{hlzi}PIc_{yul} + PIc_{hlzl}thmin + PIc_{hlzu}thmax$
wref	$wref$	Algeb	wt
PIw_yul	PIw_{yul}	Algeb	$Kpw\left(Kcc\left(Pord - Pref\right) - wref + wt\right)$
PIw_y	PIw_y	Algeb	$PIw_{hlzi}PIw_{yul} + PIw_{hlzl}thmin + PIw_{hlzu}thmax$
theta	θ	ExtAlgeb	
Pref	$Pref$	ExtAlgeb	

Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
PIc_xi	PIc_{ξ}	State	$Kic\left(Pord - Pref\right)$	
PIw_xi	PIw_{ξ}	State	$Kiw\left(Kcc\left(Pord - Pref\right) - wref + wt\right)$	
LG_y	LG_y	State	$-LG_y + 1.0PIc_y + 1.0PIw_y$	T_{θ}
wt	wt	ExtState	0	
Pord	$Pord$	ExtState	0	

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
PIc_yul	PIc_{yul}	Algeb	$Kpc\left(Pord - Pref\right) + PIc_{\xi} - PIc_{yul}$
PIc_y	PIc_y	Algeb	$PIc_{hlzi}PIc_{yul} + PIc_{hlzl}thmin + PIc_{hlzu}thmax - PIc_y$
wref	$wref$	Algeb	$-wref + wref_0$
PIw_yul	PIw_{yul}	Algeb	$Kpw\left(Kcc\left(Pord - Pref\right) - wref + wt\right) + PIw_{\xi} - PIw_{yul}$
PIw_y	PIw_y	Algeb	$PIw_{hlzi}PIw_{yul} + PIw_{hlzl}thmin + PIw_{hlzu}thmax - PIw_y$
theta	θ	ExtAlgeb	$LG_y - \theta_0$
Pref	$Pref$	ExtAlgeb	0

Services

Name	Symbol	Equation	Type
wref0	$wref0$	$wref$	PostInitService

Discretes

Name	Symbol	Type	Info
PIc_aw	aw_{PI_c}	AntiWindup	
PIc_hl	hl_{PI_c}	HardLimiter	
PIw_aw	aw_{PI_w}	AntiWindup	
PIw_hl	hl_{PI_w}	HardLimiter	
LG_lim	lim_{LG}	AntiWindupRate	Limiter in Lag

Blocks

Name	Symbol	Type	Info
PIc	PI_c	PIAWHardLimit	PI for active power diff compensation
PIw	PI_w	PIAWHardLimit	PI for speed and active power deviation
LG	LG	LagAntiWindupRate	Output lag anti-windup rate limiter

Config Fields in [WTPTA1]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

5.3.28 RenPlant

Renewable plant control group.

Common Parameters: u, name, ree

Available models: *REPCA1*

REPCA1

REPCA1: renewable energy power plant control model.

The output of the model, `Pext` and `Qext`, are the increment signals of active and reactive power for the electrical control model.

When BUS1 and BUS2 are both 0 in the DYR file (no line monitoring), the line reference is set to None and Iline evaluates to zero.

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
ree		RenExciter idx			mandatory
line		Idx of line that connect to measured bus			
busr		Optional remote bus for voltage and freq. measurement			
busf		BusFreq idx for mode 2			
VCFlag		Droop flag; 0-with droop if power factor ctrl, 1-line drop comp.		*bool*	mandatory
RefFlag		Q/V select; 0-Q control, 1-V control		*bool*	mandatory
Fflag		Frequency control flag; 0-disable, 1-enable		*bool*	mandatory
PLflag		Pline ctrl. flag; 0-disable, 1-enable	True	*bool*	
Tfltr	T_{fltr}	V or Q filter time const.	0.020		
Kp	K_p	Q proportional gain	1		
Ki	K_i	Q integral gain	0.100		
Tft	T_{ft}	Lead time constant	1		
Tfv	T_{fv}	Lag time constant	1		
Vfrz	V_{frz}	Voltage below which s2 is frozen	0.800		
Rc	R_c	Line drop compensation R			
Xc	X_c	Line drop compensation R			
Kc	K_c	Reactive power compensation gain	0		
emax	e_{max}	Upper limit on deadband output	999		
emin	e_{min}	Lower limit on deadband output	-999		
dbd1	d_{bd1}	Lower threshold for reactive power control deadband (<=0)	-0.100		
dbd2	d_{bd2}	Upper threshold for reactive power control deadband (>=0)	0.100		
Qmax	Q_{max}	Upper limit on output of V-Q control	999		
Qmin	Q_{min}	Lower limit on output of V-Q control	-999		
Kpg	K_{pg}	Proportional gain for power control	1		
Kig	K_{ig}	Integral gain for power control	0.100		
Tp	T_p	Time constant for P measurement	0.020		
fdbd1	f_{dbd1}	Lower threshold for freq. error deadband	-0.000	*p.u. (Hz)*	
fdbd2	f_{dbd2}	Upper threshold for freq. error deadband	0.000	*p.u. (Hz)*	
femax	f_{emax}	Upper limit for freq. error	0.050		

continues on next page

Table 37 – continued from previous page

Name	Symbol	Description	Default	Unit	Properties
femin	f_{emin}	Lower limit for freq. error	-0.050		
Pmax	P_{max}	Upper limit on power error (used by PI ctrl.)	999	*p.u. (MW)*	power
Pmin	P_{min}	Lower limit on power error (used by PI ctrl.)	-999	*p.u. (MW)*	power
Tg	T_g	Power controller lag time constant	0.020		
Ddn	D_{dn}	Reciprocal of droop for over-freq. conditions	10		
Dup	D_{up}	Reciprocal of droop for under-freq. conditions	10		
ue	u_e	effective online status	0		
reg		Retrieved RenGen idx			
bus		Retrieved bus idx			
bus1		Retrieved Line.bus1 idx			
bus2		Retrieved Line.bus2 idx			
r		Retrieved Line.r	1		
x		Retrieved Line.x	1		

Variables

Name	Symbol	Type	Description	Unit	Properties
s0_y	s_{0y}	State	State in lag transfer function		v_str
s1_y	s_{1y}	State	State in lag transfer function		v_str
s2_xi	$s_{2\xi}$	State	Integrator output		v_str
s3_x	s_{3x}	State	State in lead-lag		v_str
s4_y	s_{4y}	State	State in lag transfer function		v_str
s5_xi	$s_{5\xi}$	State	Integrator output		v_str
s6_y	s_{6y}	State	State in lag transfer function		v_str
Vref	$Vref$	Algeb			v_str
Qlinef	$Qlinef$	Algeb			v_str
Refsel	$Refsel$	Algeb			v_str
dbd_y	dbd_y	Algeb	Deadband type 1 output		v_str
enf	enf	Algeb	e Hardlimit output before freeze		v_str
s2_ys	s_{2ys}	Algeb	PI summation before limit		v_str
s2_y	s_{2y}	Algeb	PI output		v_str
s3_y	s_{3y}	Algeb	Output of lead-lag		v_str
ferr	$ferr$	Algeb	Frequency deviation	*p.u. (Hz)*	v_str
fdbd_y	$fdbd_y$	Algeb	Deadband type 1 output		v_str
Plant_pref	$Plant_{pref}$	Algeb	Plant P ref		v_str
Plerr	$Plerr$	Algeb	Pline error		v_str
Perr	$Perr$	Algeb	Power error before fe limits		v_str
s5_ys	s_{5ys}	Algeb	PI summation before limit		v_str
s5_y	s_{5y}	Algeb	PI output		v_str
Pext	$Pext$	ExtAlgeb	Pref from RenExciter renamed as Pext		
Qext	$Qext$	ExtAlgeb	Qref from RenExciter renamed as Qext		

continues on next page

Table 38 – continued from previous page

Name	Symbol	Type	Description	Unit	Properties
v	v	ExtAlgeb	Bus (or busr, if given) terminal voltage		
a	a	ExtAlgeb	Bus (or busr, if given) phase angle		
f	f	ExtAlgeb	Bus frequency	*p.u.*	
v1	v_1	ExtAlgeb	Voltage at Line.bus1		
v2	v_2	ExtAlgeb	Voltage at Line.bus2		
a1	a_1	ExtAlgeb	Angle at Line.bus1		
a2	a_2	ExtAlgeb	Angle at Line.bus2		

Initialization Equations

Name	Symbol	Type	Initial Value
s0_y	s_{0y}	State	$SWVC_{s0}\left(KcQline + v\right) + SWVC_{s1}Vcomp$
s1_y	s_{1y}	State	$Qline$
s2_xi	$s_{2\xi}$	State	0.0
s3_x	s_{3x}	State	s_{2y}
s4_y	s_{4y}	State	$Pline$
s5_xi	$s_{5\xi}$	State	0.0
s6_y	s_{6y}	State	s_{5y}
Vref	$Vref$	Algeb	$Vref_0$
Qlinef	$Qlinef$	Algeb	$Qline_0$
Refsel	$Refsel$	Algeb	$SWRef_{s0}\left(Qlinef - s_{1y}\right) + SWRef_{s1}\left(Vref - s_{0y}\right)$
dbd_y	dbd_y	Algeb	$1.0dbd_{dbzl}\left(Refsel - dbd_1\right) + 1.0dbd_{dbzu}\left(Refsel - dbd_2\right)$
enf	enf	Algeb	$dbd_y eHL_{zi} + eHL_{zl}emin + eHL_{zu}emax$
s2_ys	s_{2ys}	Algeb	$KpeHld$
s2_y	s_{2y}	Algeb	$Qmaxs_{2limzu} + Qmins_{2limzl} + s_{2limzi}s_{2ys}$
s3_y	s_{3y}	Algeb	s_{2y}
ferr	$ferr$	Algeb	$Freq_{ref} - f$
fdbd_y	$fdbd_y$	Algeb	$1.0fdbd_{dbzl}\left(-fdbd_1 + ferr\right) + 1.0fdbd_{dbzu}\left(-fdbd_2 + ferr\right)$
Plant_pref	$Plant_{pref}$	Algeb	$Pline_0$
Plerr	$Plerr$	Algeb	$Plant_{pref} - s_{4y}$
Perr	$Perr$	Algeb	$Ddnfdbd_y fdlt_{0z1} + Dupfdbd_y fdlt_{0z0} + PlerrSWPL_{s1}$
s5_ys	s_{5ys}	Algeb	$Kpg\left(PerrfeHL_{zi} + feHL_{zl}femin + feHL_{zu}femax\right)$
s5_y	s_{5y}	Algeb	$Pmaxs_{5limzu} + Pmins_{5limzl} + s_{5limzi}s_{5ys}$
Pext	$Pext$	ExtAlgeb	
Qext	$Qext$	ExtAlgeb	
v	v	ExtAlgeb	
a	a	ExtAlgeb	
f	f	ExtAlgeb	
v1	v_1	ExtAlgeb	
v2	v_2	ExtAlgeb	
a1	a_1	ExtAlgeb	

continues on next page

Table 39 – continued from previous page

Name	Symbol	Type	Initial Value
a2	a_2	ExtAlgeb	

Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
s0_y	s_{0y}	State	$SWVC_{s0}\left(KcQline + v\right) + SWVC_{s1}Vcomp - s_{0y}$	T_{fltr}
s1_y	s_{1y}	State	$Qline - s_{1y}$	T_{fltr}
s2_xi	$s_{2\xi}$	State	$Ki\left(eHld + 2s_{2y} - 2s_{2ys}\right)$	
s3_x	s_{3x}	State	$s_{2y} - s_{3x}$	T_{fv}
s4_y	s_{4y}	State	$Pline - s_{4y}$	T_p
s5_xi	$s_{5\xi}$	State	$Kig\left(PerrfeHL_{zi} + feHL_{zl}femin + feHL_{zu}femax + 2s_{5y} - \right.$	
s6_y	s_{6y}	State	$s_{5y} - s_{6y}$	T_g

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
Vref	$Vref$	Algeb	$-Vref + Vref_0$
Qlinef	$Qlinef$	Algeb	$Qline_0 - Qlinef$
Refsel	$Refsel$	Algeb	$-Refsel + SWRef_{s0}\left(Qlinef - s_{1y}\right) + SWRef_{s1}\left(Vref - s_{0y}\right)$
dbd_y	dbd_y	Algeb	$1.0dbd_{dbzl}\left(Refsel - dbd_1\right) + 1.0dbd_{dbzu}\left(Refsel - dbd_2\right) - dbd_y$
enf	enf	Algeb	$dbd_y eHL_{zi} + eHL_{zl} emin + eHL_{zu} emax - enf$
s2_ys	s_{2ys}	Algeb	$KpeHld + s_{2\xi} - s_{2ys}$
s2_y	s_{2y}	Algeb	$Qmaxs_{2limzu} + Qmins_{2limzl} + s_{2limzi} s_{2ys} - s_{2y}$
s3_y	s_{3y}	Algeb	$Tft\left(s_{2y} - s_{3x}\right) + Tfvs_{3x} - Tfvs_{3y} + s_{3LTz1}\left(-s_{2y} + s_{3y}\right)$
ferr	$ferr$	Algeb	$Freq_{ref} - f - ferr$
fdbd_y	$fdbd_y$	Algeb	$1.0fdbd_{dbzl}\left(-fdbd_1 + ferr\right) + 1.0fdbd_{dbzu}\left(-fdbd_2 + ferr\right) - fdbd_y$
Plant_pref	$Plant_{pref}$	Algeb	$-Plant_{pref} + Pline_0$
Plerr	$Plerr$	Algeb	$Plant_{pref} - Plerr - s_{4y}$
Perr	$Perr$	Algeb	$Ddnfdbd_y fdlt_{0z1} + Dupfdbd_y fdlt_{0z0} - Perr + PlerrSWPL_{s1}$
s5_ys	s_{5ys}	Algeb	$Kpg\left(PerrfeHL_{zi} + feHL_{zl} femin + feHL_{zu} femax\right) + s_{5\xi} - s_{5ys}$
s5_y	s_{5y}	Algeb	$Pmaxs_{5limzu} + Pmins_{5limzl} + s_{5limzi} s_{5ys} - s_{5y}$
Pext	$Pext$	ExtAlgeb	$SWF_{s1} s_{6y} ue$
Qext	$Qext$	ExtAlgeb	$s_{3y} ue$
v	v	ExtAlgeb	0
a	a	ExtAlgeb	0
f	f	ExtAlgeb	0
v1	v_1	ExtAlgeb	0
v2	v_2	ExtAlgeb	0
a1	a_1	ExtAlgeb	0
a2	a_2	ExtAlgeb	0

Services

Name	Symbol	Equation	Type
_no_line	${}_{n}o_{l}ine$	0	ConstService
Isign	I_{sign}	0	CurrentSign
Iline	I_{line}	$\frac{Isign\left(v_1 e^{ia_1} - v_2 e^{ia_2}\right)}{r+ix}$	VarService
Iline0	I_{line0}	$Iline$	ConstService
Pline	P_{line}	$Isign v_1 \text{ re}\left(\text{conj}\left(\frac{v_1 e^{ia_1} - v_2 e^{ia_2}}{r+ix}\right) e^{ia_1}\right)$	VarService
Pline0	P_{line0}	$Pline$	ConstService
Qline	Q_{line}	$Isign v_1 \text{ im}\left(\text{conj}\left(\frac{v_1 e^{ia_1} - v_2 e^{ia_2}}{r+ix}\right) e^{ia_1}\right)$	VarService
Qline0	Q_{line0}	$Qline$	ConstService
Vcomp	V_{comp}	$\left\lvert Iline\left(Rcs + iXcs\right) - ve^{ia}\right\rvert$	VarService
Vref0	V_{ref0}	$SWVC_{s0}\left(KcQline_0 + v\right) + SWVC_{s1} Vcomp$	ConstService
zf	z_f	f_{rz} Indicator $\left(v < Vfrz\right)$	VarService
eHld	e_{hld}	0	VarHold
Freq_ref	f_{ref}	1.0	ConstService

Discretes

Name	Symbol	Type	Info
SWVC	SW_{VC}	Switcher	
SWRef	SW_{Ref}	Switcher	
SWF	SW_F	Switcher	
SWPL	SW_{PL}	Switcher	
dbd_db	$db_{d^{bd}}$	DeadBand	
eHL	e_{HL}	Limiter	Hardlimit on deadband output
s2_lim	lim_{s_2}	HardLimiter	
s3_LT	LT_{s_3}	LessThan	
fdbd_db	$db_{f^{dbd}}$	DeadBand	
fdlt0	f_{dlt0}	LessThan	frequency deadband output less than zero
feHL	f_{eHL}	Limiter	Limiter for power (frequency) error
s5_lim	lim_{s_5}	HardLimiter	

Blocks

Name	Symbol	Type	Info
s0	s_0	Lag	V filter
s1	s_1	Lag	
dbd	d^{bd}	DeadBand1	
s2	s_2	PITrackAW	PI controller for eHL output
s3	s_3	LeadLag	
s4	s_4	Lag	Pline filter
fdbd	f^{dbd}	DeadBand1	frequency error deadband
s5	s_5	PITrackAW	PI for fe limiter output
s6	s_6	Lag	Output filter for Pext

Config Fields in [REPCA1]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)
kqs	K_{qs}	2	Tracking gain for reactive power PI controller	
ksg	K_{sg}	2	Tracking gain for active power PI controller	
freeze	f_{rz}	1	Voltage dip freeze flag; 1-enable, 0-disable	

5.3.29 RenTorque

Renewable torque (Pref) controller.

Common Parameters: u, name

Available models: *WTTQA1*

WTTQA1

Wind turbine generator torque (Pref) model.

PI state freeze following voltage dip has not been implemented.

Resets *wg* in *REECA1* model to 1.0 when torque model is connected. This effectively ignores *PFLAG* of *REECA1*.

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
rep		RenPitch controller idx			mandatory
Kip	K_{ip}	Pref-control integral gain	0.100	*p.u.*	
Kpp	K_{pp}	Pref-control proportional gain	0	*p.u.*	
Tp	T_p	Pe sensing time const.	0.050	*s*	
Twref	T_{wref}	Speed reference time const.	30	*s*	
Temax	T_{emax}	Max. electric torque	1.200	*p.u.*	power
Temin	T_{emin}	Min. electric torque	0	*p.u.*	power
Tflag		Tflag; 1-power error, 0-speed error		*bool*	mandatory
p1	p_1	Active power point 1	0.200	*p.u.*	power
sp1	s_{p1}	Speed power point 1	0.580	*p.u.*	
p2	p_2	Active power point 2	0.400	*p.u.*	power
sp2	s_{p2}	Speed power point 2	0.720	*p.u.*	
p3	p_3	Active power point 3	0.600	*p.u.*	power
sp3	s_{p3}	Speed power point 3	0.860	*p.u.*	
p4	p_4	Active power point 4	0.800	*p.u.*	power
sp4	s_{p4}	Speed power point 4	1	*p.u.*	
Tn	T_n	Turbine rating. Use Sn from gov if none.	nan	*MVA*	
ue	u_e	effective online status	0		
rea			0		
rego			0		
ree			0		
reg			0		
Sngo	$S_{n,go}$		0		
w0	ω_0		0		

Variables

Name	Symbol	Type	Description	Unit	Properties
s1_y	s_{1y}	State	State in lag transfer function		v_str
s2_y	s_{2y}	State	State in lag transfer function		v_str
PI_xi	π_{ξ}	State	Integrator output		v_str
wg	wg	ExtState			v_str,v_setter
wt	wt	ExtState			v_str,v_setter
s3_y	s_{3y}	ExtState			v_str,v_setter
fPe_y	fPe_y	Algeb	Output of piecewise		v_str
Tsel	$Tsel$	Algeb	Output after Tflag selector		v_str
PI_yul	π_{yul}	Algeb			v_str
PI_y	π_y	Algeb	PI output		v_str
Pe	Pe	ExtAlgeb			
wr0	wr_0	ExtAlgeb	Retrieved initial w0 from RenGovernor		v_str,v_setter
wge	wge	ExtAlgeb			v_str,v_setter
Pref	$Pref$	ExtAlgeb			v_str,v_setter

Initialization Equations

Nam	Sym-bol	Type	Initial Value
s1_y	s_{1y}	State	$1.0Pe$
s2_y	s_{2y}	State	$1.0fPe_y$
PI_xi	π_ξ	State	$\frac{Pref_0}{fPe_y}$
wg	wg	ExtSta	fPe_y
wt	wt	ExtSta	fPe_y
s3_y	s_{3y}	ExtSta	$\frac{Pref_0}{wg}$
fPe_y	fPe_y	Al-geb	FixPiecewise $((sp_1,\ p_1 \geq s_{1y}), (kp_1(-p_1 + s_{1y}) + sp_1,\ p_2 \geq s_{1y}), (kp_2(-p_2 + s_{1y}) + sp_2,\ p_3 \geq$
Tsel	$Tsel$	Al-geb	$SWT_{s0}(s_{2y} - wg) + \frac{SWT_{s1}(Pe - Pref_0)}{wg}$
PI_yu	π_{yul}	Al-geb	$KppTsel + \frac{Pref_0}{fPe_y}$
PI_y	π_y	Al-geb	$\pi_{hlzi}\pi_{yul} + \pi_{hlzl}Temin + \pi_{hlzu}Temax$
Pe	Pe	Ex-tAl-geb	
wr0	wr_0	Ex-tAl-geb	fPe_y
wge	wge	Ex-tAl-geb	1.0
Pref	$Pref$	Ex-tAl-geb	$\pi_y wg$

Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
s1_y	s_{1y}	State	$1.0Pe - s_{1y}$	T_p
s2_y	s_{2y}	State	$1.0fPe_y - s_{2y}$	T_{wref}
PI_xi	π_ξ	State	$KipTsel$	
wg	wg	ExtState	0	
wt	wt	ExtState	0	
s3_y	s_{3y}	ExtState	0	

Algebraic Equations

Nam	Sym-bol	Type	RHS of Equation "0 = g(x, y)"
fPe_y	fPe_y	Al-geb	$-fPe_y$+FixPiecewise $((sp_1,\ p_1 \geq s_{1y}), (kp_1(-p_1 + s_{1y}) + sp_1,\ p_2 \geq s_{1y}), (kp_2(-p_2 + s_{1y}) + s$
Tsel	$Tsel$	Al-geb	$SWT_{s0}(s_{2y} - wg) + \frac{SWT_{s1}(Pe - Pref_0)}{wg} - Tsel$
PI_yu	π_{yul}	Al-geb	$KppTsel + \pi_\xi - \pi_{yul}$
PI_y	π_y	Al-geb	$\pi_{hlzi}\pi_{yul} + \pi_{hlzl}Temin + \pi_{hlzu}Temax - \pi_y$
Pe	Pe	Ex-tAl-geb	0
wr0	wr_0	Ex-tAl-geb	$fPe_y - w_0$
wge	wge	Ex-tAl-geb	$1 - fPe_y$
Pref	$Pref$	Ex-tAl-geb	$\pi_y wg - \frac{Pref_0}{wge}$

Services

Name	Symbol	Equation	Type
kp1	k_{p1}	$\frac{-sp_1+sp_2}{-p_1+p_2}$	ConstService
kp2	k_{p2}	$\frac{-sp_2+sp_3}{-p_2+p_3}$	ConstService
kp3	k_{p3}	$\frac{-sp_3+sp_4}{-p_3+p_4}$	ConstService

Discretes

Name	Symbol	Type	Info
SWT	SW_T	Switcher	
PI_aw	aw_{PI}	AntiWindup	
PI_hl	hl_{PI}	HardLimiter	

Blocks

Name	Symbol	Type	Info
s1	s_1	Lag	Pe filter
fPe	f_{Pe}	Piecewise	Piecewise Pe to wref mapping
s2	s_2	Lag	speed filter
PI	PI	PIAWHardLimit	PI controller

Config Fields in [WTTQA1]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

5.3.30 StaticACDC

AC DC device for power flow

Common Parameters: u, name

Available models: *VSCShunt*

VSCShunt

Data for VSC Shunt in power flow

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
bus		idx of connected bus			mandatory
node1		Node 1 index			mandatory
node2		Node 2 index			mandatory
Vn	V_n	AC voltage rating	110		non_zero
Vdcn1	V_{dcn1}	DC voltage rating on node 1	100	*kV*	non_zero
Vdcn2	V_{dcn2}	DC voltage rating on node 2	100	*kV*	non_zero
Idcn	I_{dcn}	DC current rating	1	*kA*	non_zero
ue	u_e	effective online status	0		
rsh	r_{sh}	AC interface resistance	0.003	*ohm*	z
xsh	x_{sh}	AC interface reactance	0.060	*ohm*	z
control		Control method: 0-PQ, 1-PV, 2-vQ or 3-vV			mandatory
v0		AC voltage setting (PV or vV) or initial guess (PQ or vQ)	1		
p0		AC active power setting	0	*pu*	
q0		AC reactive power setting	0	*pu*	
vdc0	v_{dc0}	DC voltage setting	1	*pu*	
k0		Loss coefficient - constant	0		
k1		Loss coefficient - linear	0		
k2		Loss coefficient - quadratic	0		
droop		Enable dc voltage droop control	0	*boolean*	
K		Droop coefficient	0		
vhigh		Upper voltage threshold in droop control	9999	*pu*	
vlow		Lower voltage threshold in droop control	0	*pu*	
vshmax		Maximum ac interface voltage	1.100	*pu*	
vshmin		Minimum ac interface voltage	0.900	*pu*	
Ishmax		Maximum ac current	2	*pu*	

Variables

Name	Symbol	Type	Description	Unit	Properties
ash	ash	Algeb	voltage phase behind the transformer	*rad*	v_str
vsh	vsh	Algeb	voltage magnitude behind transformer	*p.u.*	v_str
psh	psh	Algeb	active power injection into VSC	*p.u.*	v_str
qsh	qsh	Algeb	reactive power injection into VSC		v_str
pdc	pdc	Algeb	DC power injection		v_str
a	a	ExtAlgeb	AC bus voltage phase		
v	v	ExtAlgeb	AC bus voltage magnitude		
v1	v_1	ExtAlgeb	DC node 1 voltage		
v2	v_2	ExtAlgeb	DC node 2 voltage		

Initialization Equations

Name	Symbol	Type	Initial Value
ash	ash	Algeb	a
vsh	vsh	Algeb	v_0
psh	psh	Algeb	$p_0\left(mode_{s0} + mode_{s1}\right)$
qsh	qsh	Algeb	$q_0\left(mode_{s0} + mode_{s2}\right)$
pdc	pdc	Algeb	0
a	a	ExtAlgeb	
v	v	ExtAlgeb	
v1	v_1	ExtAlgeb	
v2	v_2	ExtAlgeb	

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
ash	ash	Algeb	$-psh+ue\left(-bshvvsh\sin\left(a-ash\right)+gshv^{2}-gshvvsh\cos\left(a-ash\right)\right)$
vsh	vsh	Algeb	$-qsh+ue\left(-bshv^{2}+bshvvsh\cos\left(a-ash\right)-gshvvsh\sin\left(a-ash\right)\right)$
psh	psh	Algeb	$ue\left(mode_{s0}+mode_{s1}\right)\left(p_{0}-psh\right)+ue\left(mode_{s2}+mode_{s3}\right)\left(v_{1}-v_{2}-vdc_{0}\right)$
qsh	qsh	Algeb	$ue\left(mode_{s0}+mode_{s2}\right)\left(q_{0}-qsh\right)+ue\left(mode_{s1}+mode_{s3}\right)\left(-v+v_{0}\right)$
pdc	pdc	Algeb	$pdc+ue\left(bshvvsh\sin\left(a-ash\right)-gshvvsh\cos\left(a-ash\right)+gshvsh^{2}\right)$
a	a	ExtAlgeb	$-psh$
v	v	ExtAlgeb	$-qsh$
v1	v_{1}	ExtAlgeb	$-\frac{pdc}{v_{1}-v_{2}}$
v2	v_{2}	ExtAlgeb	$\frac{pdc}{v_{1}-v_{2}}$

Services

Name	Symbol	Equation	Type
gsh	g_{sh}	$\frac{rsh}{rsh^{2}+xsh^{2}}$	ConstService
bsh	b_{sh}	$-\frac{xsh}{rsh^{2}+xsh^{2}}$	ConstService

Discretes

Name	Symbol	Type	Info
mode	$mode$	Switcher	

Config Fields in [VSCShunt]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

5.3.31 StaticGen

Static generator group.

Static generators include PV and Slack, which are used to impose algebraic equations. Static generators are used primarily for power flow.

Static generators do not have the modeling details for stability simulation. Although some of them can stay for time-domain simulation, most of them should be substituted by dynamic generators, including synchronous generators and inverter-based resources upon TDS initialization.

The substitution is done by setting the `gen` field of a dynamic generator to refer to the static generator. To replace one StaticGen by multiple dynamic generators, see the notes in *SynGen*. Generators connected to the same bus need to have their *gammap* and *gammaq*, respectively, summed up to be exactly 1.0.

TDS initialization will ensure that the dynamic generators impose the same amount of power as the static generator. At the end of initialization, *StaticGen*'s that have been substituted will have their connectivity status `u` changed to `0`.

Common Parameters: u, name, bus, Sn, Vn, p0, q0, ra, xs, subidx

Common Variables: q, a, v

Available models: *PV*, *Slack*

PV

Static PV generator with reactive power limit checking and PV-to-PQ conversion.

pv2pq = 1 turns on the conversion. It starts from iteration *min_iter* or when the convergence error drops below *err_tol*.

The PV-to-PQ conversion first ranks the reactive violations. A maximum number of *npv2pq* PVs above the upper limit, and a maximum of *npv2pq* PVs below the lower limit will be converted to PQ, which sets the reactive power to *pmax* or *pmin*.

If *pv2pq* is *1* (enabled) and *npv2pq* is *0*, heuristics will be used to determine the number of PVs to be converted for each iteration.

This conversion also applies during Continuation Power Flow (CPF). Pass `config_option=["PV.pv2pq=1"]` to `andes.load()` to enforce reactive power limits along the nose curve.

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
Sn	S_n	Power rating	100		non_zero
Vn	V_n	AC voltage rating	110		non_zero
subidx		index for generators on the same bus			
bus		idx of the installed bus			mandatory
busr		bus idx for remote voltage control			
p0	p_0	active power set point in system base	0	*p.u.*	
q0	q_0	reactive power set point in system base	0	*p.u.*	
pmax	p_{max}	maximum active power in system base	999	*p.u.*	
pmin	p_{min}	minimum active power in system base	-1	*p.u.*	
qmax	q_{max}	maximim reactive power in system base	999	*p.u.*	
qmin	q_{min}	minimum reactive power in system base	-999	*p.u.*	
v0	v_0	voltage set point	1		
vmax	v_{max}	maximum voltage voltage	1.400		
vmin	v_{min}	minimum allowed voltage	0.600		
ra	r_a	armature resistance	0		
xs	x_s	armature reactance	0.300		
wmod	w_{mod}	Machine ctrl. mode [PSS/E]	0	*int*	
ue	u_e	effective online status	0		
busv0	V_{0bus}		0		

Variables

Name	Symbol	Type	Description	Unit	Properties
q	q	Algeb	actual reactive power generation	*p.u.*	v_str
a	a	ExtAlgeb			
v	v	ExtAlgeb			v_str,v_setter

Initialization Equations

Name	Symbol	Type	Initial Value
q	q	Algeb	$q_0 u$
a	a	ExtAlgeb	
v	v	ExtAlgeb	$busv_0 \left(1 - u\right) + uv_0$

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
q	q	Algeb	$ue\left(qlim_{zi}\left(-v+v_0\right)+qlim_{zl}\left(-q+qmin\right)+qlim_{zu}\left(-q+qmax\right)\right)$
a	a	ExtAlgeb	$-pue$
v	v	ExtAlgeb	$-que$

Services

Name	Symbol	Equation	Type
p	p	p_0	ConstService

Discretes

Name	Symbol	Type	Info
qlim	$qlim$	SortedLimiter	

Config Fields in [PV]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)
pv2pq	z_{pv2pq}	0	convert PV to PQ in PFlow at Q limits	(0, 1)
npv2pq	n_{pv2pq}	0	max. # of conversion each iteration, 0 - auto	>=0
min_iter	sw_{iter}	2	iteration number starting from which to enable switching	int
err_tol	ϵ_{tol}	0.010	iteration error below which to enable switching	float
abs_violation		1	use absolute (1) or relative (0) limit violation	(0, 1)

Slack

Slack generator.

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
Sn	S_n	Power rating	100		non_zero
Vn	V_n	AC voltage rating	110		non_zero
subidx		index for generators on the same bus			
bus		idx of the installed bus			mandatory
busr		bus idx for remote voltage control			
p0	p_0	active power set point in system base	0	*p.u.*	
q0	q_0	reactive power set point in system base	0	*p.u.*	
pmax	p_{max}	maximum active power in system base	999	*p.u.*	
pmin	p_{min}	minimum active power in system base	-1	*p.u.*	
qmax	q_{max}	maximim reactive power in system base	999	*p.u.*	
qmin	q_{min}	minimum reactive power in system base	-999	*p.u.*	
v0	v_0	voltage set point	1		
vmax	v_{max}	maximum voltage voltage	1.400		
vmin	v_{min}	minimum allowed voltage	0.600		
ra	r_a	armature resistance	0		
xs	x_s	armature reactance	0.300		
wmod	w_{mod}	Machine ctrl. mode [PSS/E]	0	*int*	
a0	θ_0	reference angle set point	0		
ue	u_e	effective online status	0		
busv0	V_{0bus}		0		
busa0	θ_{0bus}		0		

Variables

Name	Symbol	Type	Description	Unit	Properties
q	q	Algeb	actual reactive power generation	*p.u.*	v_str
p	p	Algeb	actual active power generation	*p.u.*	v_str
a	a	ExtAlgeb			v_str,v_setter
v	v	ExtAlgeb			v_str,v_setter

Initialization Equations

Name	Symbol	Type	Initial Value
q	q	Algeb	$q_0 u$
p	p	Algeb	$p_0 u$
a	a	ExtAlgeb	$a_0 u + busa_0 (1 - u)$
v	v	ExtAlgeb	$busv_0 (1 - u) + uv_0$

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
q	q	Algeb	$ue (qlim_{zi} (-v + v_0) + qlim_{zl} (-q + qmin) + qlim_{zu} (-q + qmax))$
p	p	Algeb	$ue (plim_{zi} (-a + a_0) + plim_{zl} (-p + pmin) + plim_{zu} (-p + pmax))$
a	a	ExtAlgeb	$-pue$
v	v	ExtAlgeb	$-que$

Discretes

Name	Symbol	Type	Info
qlim	$qlim$	SortedLimiter	
plim	$plim$	SortedLimiter	

Config Fields in [Slack]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)
pv2pq	z_{pv2pq}	0	convert PV to PQ in PFlow at Q limits	(0, 1)
npv2pq	n_{pv2pq}	0	max. # of conversion each iteration, 0 - auto	>=0
min_iter	sw_{iter}	2	iteration number starting from which to enable switching	int
err_tol	ϵ_{tol}	0.010	iteration error below which to enable switching	float
abs_violation		1	use absolute (1) or relative (0) limit violation	(0, 1)
av2pv	z_{av2pv}	0	convert Slack to PV in PFlow at P limits	(0, 1)

5.3.32 StaticLoad

Static load group.

Common Parameters: u, name

Available models: *PQ*

PQ

PQ load model.

Implements an automatic pq2z conversion during power flow when the voltage is outside [vmin, vmax]. The conversion can be turned off by setting *pq2z* to 0 in the Config file.

Before time-domain simulation, PQ load will be converted to impedance, current source, and power source based on the weights in the Config file.

Weights (p2p, p2i, p2z) corresponds to the weights for constant power, constant current and constant impedance. p2p, p2i and p2z must be in decimal numbers and sum up exactly to 1. The same rule applies to (q2q, q2i, q2z).

To alter the PQ load in terms of power during simulation, one needs to set the conversion weights to preserve the constant power portion. For example, the PQ can remain as constant power load by setting

```
ss.PQ.config.p2p = 1.0
ss.PQ.config.p2i = 0
ss.PQ.config.p2z = 0

ss.PQ.config.q2q = 1.0
ss.PQ.config.q2i = 0
ss.PQ.config.q2z = 0
```

Then, the constant power portion can be altered by changing the `Ppf` and `Qpf` constants for active power and reactive power.

The equivalent constant current components are in constants `Ipeq` and `Iqeq` for active and reactive current, and the equivalent impedances are in `Req` and `Xeq`.

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
bus		linked bus idx			mandatory
Vn	V_n	AC voltage rating	110	kV	non_zero
p0	p_0	active power load in system base	0	*p.u.*	
q0	q_0	reactive power load in system base	0	*p.u.*	
vmax	v_{max}	max voltage before switching to impedance	1.200		
vmin	v_{min}	min voltage before switching to impedance	0.800		
owner		owner idx			
ue	u_e	effective online status	0		

Variables

Name	Symbol	Type	Description	Unit	Properties
a	a	ExtAlgeb			
v	v	ExtAlgeb			

Initialization Equations

Name	Symbol	Type	Initial Value
a	a	ExtAlgeb	
v	v	ExtAlgeb	

Algebraic Equations

Nam	Symbol	Type	RHS of Equation "0 = g(x, y)"	
a	a	ExtAlgeb	$ue\left(Ipeq\gamma_{p2i}v + Ppf\gamma_{p2p} + Req\gamma_{p2z}v^2\right)$ Indicator $(t_{dae} \geq 0)$ $ue\left(Rlbv^2vcmp_{zl} + Rubv^2vcmp_{zu} + p_0vcmp_{zi}\right)$ Indicator $(t_{dae} < 0)$	$+$
v	v	ExtAlgeb	$ue\left(Iqeq\gamma_{q2i}v + Qpf\gamma_{q2q} + Xeq\gamma_{q2z}v^2\right)$ Indicator $(t_{dae} \geq 0)$ $ue\left(Xlbv^2vcmp_{zl} + Xubv^2vcmp_{zu} + q_0vcmp_{zi}\right)$ Indicator $(t_{dae} < 0)$	$+$

Services

Name	Symbol	Equation	Type
Rub	R_{ub}	$\frac{p_0}{vmax^2}$	ConstService
Xub	X_{ub}	$\frac{q_0}{vmax^2}$	ConstService
Rlb	R_{lb}	$\frac{p_0}{vmin^2}$	ConstService
Xlb	X_{lb}	$\frac{q_0}{vmin^2}$	ConstService
Ppf	P_{pf}	$Rlbv_0^2vcmp_{zl} + Rubv_0^2vcmp_{zu} + p_0vcmp_{zi}$	ConstService
Qpf	Q_{pf}	$Xlbv_0^2vcmp_{zl} + Xubv_0^2vcmp_{zu} + q_0vcmp_{zi}$	ConstService
Req	R_{eq}	$\frac{Ppf}{v_0^2}$	ConstService
Xeq	X_{eq}	$\frac{Qpf}{v_0^2}$	ConstService
Ipeq	I_{peq}	$\frac{Ppf}{v_0}$	ConstService
Iqeq	I_{qeq}	$\frac{Qpf}{v_0}$	ConstService

Discretes

Name	Symbol	Type	Info
vcmp	$vcmp$	Limiter	

Config Fields in [PQ]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)
pq2z	z_{pq2z}	1	pq2z conversion if out of voltage limits	(0, 1)
p2p	γ_{p2p}	0	P constant power percentage for TDS. Must have (p2p+p2i+p2z)=1	float
p2i	γ_{p2i}	0	P constant current percentage	float
p2z	γ_{p2z}	1	P constant impedance percentage	float
q2q	γ_{q2q}	0	Q constant power percentage for TDS. Must have (q2q+q2i+q2z)=1	float
q2i	γ_{q2i}	0	Q constant current percentage	float
q2z	γ_{q2z}	1	Q constant impedance percentage	float

5.3.33 StaticShunt

Static shunt compensator group.

Common Parameters: u, name

Available models: *Shunt*, *ShuntTD*, *ShuntSw*

Shunt

Phasor-domain shunt compensator Model.

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
bus		idx of connected bus			mandatory
Sn	S_n	Power rating	100		non_zero
Vn	V_n	AC voltage rating	110		non_zero
g	g	shunt conductance (real part)	0		y
b	b	shunt susceptance (positive as capacitive)	0		y
fn	f_n	rated frequency	60		
ue	u_e	effective online status	0		

Variables

Name	Symbol	Type	Description	Unit	Properties
a	a	ExtAlgeb			
v	v	ExtAlgeb			

Initialization Equations

Name	Symbol	Type	Initial Value
a	a	ExtAlgeb	
v	v	ExtAlgeb	

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
a	a	ExtAlgeb	$guev^2$
v	v	ExtAlgeb	$-buev^2$

Config Fields in [Shunt]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

ShuntTD

Static shunt model with inverse transformation from phasor to time-domain.

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
bus		idx of connected bus			mandatory
Sn	S_n	Power rating	100		non_zero
Vn	V_n	AC voltage rating	110		non_zero
g	g	shunt conductance (real part)	0		y
b	b	shunt susceptance (positive as capacitive)	0		y
fn	f_n	rated frequency	60		
ue	u_e	effective online status	0		

Variables

Name	Symbol	Type	Description	Unit	Properties
vta	vta	Algeb			v_str
vtb	vtb	Algeb			v_str
vtc	vtc	Algeb			v_str
a	a	ExtAlgeb			
v	v	ExtAlgeb			

Initialization Equations

Name	Symbol	Type	Initial Value
vta	vta	Algeb	$\frac{\sqrt{3}v\cos\left(a+2\pi f_{sys}t_{dae}\right)}{3}$
vtb	vtb	Algeb	$-\frac{\sqrt{3}v\cos\left(a+2\pi f_{sys}t_{dae}+\frac{\pi}{3}\right)}{3}$
vtc	vtc	Algeb	$-\frac{\sqrt{3}v\sin\left(a+2\pi f_{sys}t_{dae}+\frac{\pi}{6}\right)}{3}$
a	a	ExtAlgeb	
v	v	ExtAlgeb	

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
vta	vta	Algeb	$\frac{\sqrt{3}v\cos\left(a+2\pi f_{sys}t_{dae}\right)}{3} - vta$
vtb	vtb	Algeb	$-\frac{\sqrt{3}v\cos\left(a+2\pi f_{sys}t_{dae}+\frac{\pi}{3}\right)}{3} - vtb$
vtc	vtc	Algeb	$-\frac{\sqrt{3}v\sin\left(a+2\pi f_{sys}t_{dae}+\frac{\pi}{6}\right)}{3} - vtc$
a	a	ExtAlgeb	$guev^2$
v	v	ExtAlgeb	$-buev^2$

Config Fields in [ShuntTD]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

ShuntSw

Switched Shunt Model.

Parameters *gs*, *bs* and *bs* must be entered in string literals, comma-separated. They need to have the same length.

For example, in the excel file, one can put

```
gs = [0, 0]
bs = [0.2, 0.2]
ns = [2, 4]
```

To use individual shunts as fixed shunts, set the corresponding *ns = 0* or *ns = [0]*.

The effective shunt susceptances and conductances are stored in services *beff* and *geff*.

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
bus		idx of connected bus			mandatory
Sn	S_n	Power rating	100		non_zero
Vn	V_n	AC voltage rating	110		non_zero
g	g	shunt conductance (real part)	0		y
b	b	shunt susceptance (positive as capacitive)	0		y
fn	f_n	rated frequency	60		
gs		a list literal of switched conductances blocks	0	*p.u.*	y
bs		a list literal of switched susceptances blocks	0	*p.u.*	y
ns		a list literal of the element numbers in each switched block	[0]		
vref		voltage reference	1	*p.u.*	non_zero,non_negative
dv		voltage error deadband	0.050	*p.u.*	non_zero,non_negative
dt		delay before two consecutive switching	30	*seconds*	non_negative
ue	u_e	effective online status	0		

Variables

Name	Symbol	Type	Description	Unit	Properties
a	a	ExtAlgeb			
v	v	ExtAlgeb			

Initialization Equations

Name	Symbol	Type	Initial Value
a	a	ExtAlgeb	
v	v	ExtAlgeb	

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
a	a	ExtAlgeb	$geffuev^2$
v	v	ExtAlgeb	$-beffuev^2$

Services

Name	Symbol	Equation	Type
vlo	v_{lo}	$-dv + vref$	ConstService
vup	v_{up}	$dv + vref$	ConstService

Discretes

Name	Symbol	Type	Info
adj	adj	ShuntAdjust	shunt adjuster

Config Fields in [ShuntSw]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)
min_iter	sw_{iter}	2	iteration number starting from which to enable switching	int
err_tol	ϵ_{tol}	0.010	iteration error below which to enable switching	float

5.3.34 SynGen

Synchronous generator group.

SynGen replaces StaticGen upon the initialization of dynamic studies. SynGen and inverter-based resources contain parameters `gammap` and `gammaq` for splitting the initial power of a StaticGen into multiple dynamic ones.

`gammap`, for example, is the active power ratio of the dynamic generator to the static one. If a StaticGen is supposed to be replaced by one SynGen, the `gammap` and `gammaq` should both be `1`.

It is critical to ensure that `gammap` and `gammaq`, respectively, of all dynamic power sources sum up to 1.0. Otherwise, the initial power injections imposed by dynamic sources will differ from the static ones. The initialization will then fail with mismatches power injection equations corresponding to bus `a` and `v`.

Common Parameters: u, name, bus, gen, Sn, Vn, fn, M, D, subidx

Common Variables: omega, delta

Available models: *GENCLS*, *GENROU*, *PLBVFU1*

GENCLS

Classical generator model.

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
bus		interface bus id			mandatory
gen		static generator index			mandatory
coi		center of inertia index			
coi2		center of inertia index			
Sn	S_n	Power rating	100	*MVA*	
Vn	V_n	AC voltage rating	110		
fn	f	rated frequency	60		
D	D	Damping coefficient	0		power
M	M	machine start up time (2H)	6		non_zero,non_negative,power
ra	r_a	armature resistance	0		z
xl	x_l	leakage reactance	0		z
xd1	x'_d	d-axis transient reactance	0.302		z
kp	k_p	active power feedback gain	0		
kw	k_w	speed feedback gain	0		
S10	$S_{1.0}$	first saturation factor	0		
S12	$S_{1.2}$	second saturation factor	1		
gammap	γ_P	P ratio of linked static gen	1		
gammaq	γ_Q	Q ratio of linked static gen	1		
ue	u_e	effective online status	0		
subidx		Generator idx in plant; only used by PSS/E data	0		

Variables

Name	Symbol	Type	Description	Unit	Properties
delta	δ	State	rotor angle	*rad*	v_str
omega	ω	State	rotor speed	*pu (Hz)*	v_str
Id	Id	Algeb	d-axis current		v_str
Iq	Iq	Algeb	q-axis current		v_str
vd	vd	Algeb	d-axis voltage		v_str
vq	vq	Algeb	q-axis voltage		v_str
tm	tm	Algeb	mechanical torque		v_str
te	te	Algeb	electric torque		v_str
vf	vf	Algeb	excitation voltage	*pu*	v_str
XadIfd	$XadIfd$	Algeb	d-axis armature excitation current	*p.u (kV)*	v_str
Pe	Pe	Algeb	active power injection		v_str
Qe	Qe	Algeb	reactive power injection		v_str
psid	$psid$	Algeb	d-axis flux		v_str
psiq	$psiq$	Algeb	q-axis flux		v_str
a	a	ExtAlgeb	Bus voltage phase angle		
v	v	ExtAlgeb	Bus voltage magnitude		

Initialization Equations

Name	Symbol	Type	Initial Value
delta	δ	State	δ_0
omega	ω	State	u
Id	Id	Algeb	$Id_0 u$
Iq	Iq	Algeb	$Iq_0 u$
vd	vd	Algeb	uvd_0
vq	vq	Algeb	uvq_0
tm	tm	Algeb	tm_0
te	te	Algeb	$tm_0 u$
vf	vf	Algeb	uvf_0
XadIfd	$XadIfd$	Algeb	uvf_0
Pe	Pe	Algeb	$u\left(Id_0 vd_0 + Iq_0 vq_0\right)$
Qe	Qe	Algeb	$u\left(Id_0 vq_0 - Iq_0 vd_0\right)$
psid	$psid$	Algeb	$psid_0 u$
psiq	$psiq$	Algeb	$psiq_0 u$
a	a	ExtAlgeb	
v	v	ExtAlgeb	

Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
delta	δ	State	$2\pi fnue\left(\omega - 1\right)$	
omega	ω	State	$ue\left(-D\left(\omega - 1\right) - te + tm\right)$	M

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
Id	Id	Algeb	$Idxq + psid - vf$
Iq	Iq	Algeb	$Iqxq + psiq$
vd	vd	Algeb	$-uev\sin\left(a - \delta\right) - vd$
vq	vq	Algeb	$uev\cos\left(a - \delta\right) - vq$
tm	tm	Algeb	$-tm + tm_0$
te	te	Algeb	$-te + ue\left(-Idpsiq + Iqpsid\right)$
vf	vf	Algeb	$uevf_0 - vf$
XadIfd	$XadIfd$	Algeb	$-XadIfd + uevf_0$
Pe	Pe	Algeb	$-Pe + ue\left(Idvd + Iqvq\right)$
Qe	Qe	Algeb	$-Qe + ue\left(Idvq - Iqvd\right)$
psid	$psid$	Algeb	$-psid + ue\left(Iqra + vq\right)$
psiq	$psiq$	Algeb	$psiq + ue\left(Idra + vd\right)$
a	a	ExtAlgeb	$-ue\left(Idvd + Iqvq\right)$
v	v	ExtAlgeb	$-ue\left(Idvq - Iqvd\right)$

Services

Name	Symbol	Equation	Type
p0	P_0	$gammapp0s$	ConstService
q0	Q_0	$gammaqq0s$	ConstService
_V	V_c	ve^{ia}	ConstService
_S	S	$p_0 - iq_0$	ConstService
_I	I_c	$\frac{S}{\mathrm{conj}\left(V\right)}$	ConstService
_E	E	${}_I\left(ra + ixq\right) + {}_V$	ConstService
_deltac	δ_c	$\log\left(\frac{E}{\left\lvert E\right\rvert}\right)$	ConstService
delta0	δ_0	$u\,\mathrm{im}\left({}_d eltac\right)$	ConstService
vdq	V_{dq}	${}_V ue^{-{}_d eltac + 0.5i\pi}$	ConstService
Idq	I_{dq}	${}_I ue^{-{}_d eltac + 0.5i\pi}$	ConstService
Id0	I_{d0}	$\mathrm{re}\left(Idq\right)$	ConstService
Iq0	I_{q0}	$\mathrm{im}\left(Idq\right)$	ConstService
vd0	V_{d0}	$\mathrm{re}\left(vdq\right)$	ConstService
vq0	V_{q0}	$\mathrm{im}\left(vdq\right)$	ConstService
tm0	τ_{m0}	$u\left(Id_0\left(Id_0 ra + vd_0\right) + Iq_0\left(Iq_0 ra + vq_0\right)\right)$	ConstService
psid0	ψ_{d0}	$Iq_0 rau + vq_0$	ConstService
psiq0	ψ_{q0}	$-Id_0 rau - vd_0$	ConstService
vf0	v_{f0}	$Id_0 xq + Iq_0 ra + vq_0$	ConstService

Setpoints

Name	Target	Info
pref	tm0	
vref	vf0	

Config Fields in [GENCLS]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)
vf_lower		1	lower limit for vf warning	
vf_upper		5	upper limit for vf warning	

GENROU

Round rotor generator with quadratic saturation.

Notes

Parameters:

- `xd2` and `xq2` must be equal to pass initialization.

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
bus		interface bus id			mandatory
gen		static generator index			mandatory
coi		center of inertia index			
coi2		center of inertia index			
Sn	S_n	Power rating	100	*MVA*	
Vn	V_n	AC voltage rating	110		
fn	f	rated frequency	60		
D	D	Damping coefficient	0		power
M	M	machine start up time (2H)	6		non_zero,non_negative,power
ra	r_a	armature resistance	0		z
xl	x_l	leakage reactance	0		z
xd1	x'_d	d-axis transient reactance	0.302		z
kp	k_p	active power feedback gain	0		
kw	k_w	speed feedback gain	0		
S10	$S_{1.0}$	first saturation factor	0		
S12	$S_{1.2}$	second saturation factor	1		
gammap	γ_P	P ratio of linked static gen	1		
gammaq	γ_Q	Q ratio of linked static gen	1		
xd	x_d	d-axis synchronous reactance	1.900		z
xq	x_q	q-axis synchronous reactance	1.700		z
xd2	x''_d	d-axis sub-transient reactance	0.300		z
xq1	x'_q	q-axis transient reactance	0.500		z
xq2	x''_q	q-axis sub-transient reactance	0.300		z
Td10	T'_{d0}	d-axis transient time constant	8		
Td20	T''_{d0}	d-axis sub-transient time constant	0.040		
Tq10	T'_{q0}	q-axis transient time constant	0.800		
Tq20	T''_{q0}	q-axis sub-transient time constant	0.020		
ue	u_e	effective online status	0		

continues on next page

Table 40 – continued from previous page

Name	Symbol	Description	Default	Unit	Properties
subidx		Generator idx in plant; only used by PSS/E data	0		

Variables

Name	Symbol	Type	Description	Unit	Properties
delta	δ	State	rotor angle	*rad*	v_str
omega	ω	State	rotor speed	*pu (Hz)*	v_str
e1q	$e1q$	State	q-axis transient voltage		v_str
e1d	$e1d$	State	d-axis transient voltage		v_str
e2d	$e2d$	State	d-axis sub-transient voltage		v_str
e2q	$e2q$	State	q-axis sub-transient voltage		v_str
Id	Id	Algeb	d-axis current		v_str
Iq	Iq	Algeb	q-axis current		v_str
vd	vd	Algeb	d-axis voltage		v_str
vq	vq	Algeb	q-axis voltage		v_str
tm	tm	Algeb	mechanical torque		v_str
te	te	Algeb	electric torque		v_str
vf	vf	Algeb	excitation voltage	*pu*	v_str
XadIfd	$XadIfd$	Algeb	d-axis armature excitation current	*p.u (kV)*	v_str
Pe	Pe	Algeb	active power injection		v_str
Qe	Qe	Algeb	reactive power injection		v_str
psid	$psid$	Algeb	d-axis flux		v_str
psiq	$psiq$	Algeb	q-axis flux		v_str
psi2q	$psi2q$	Algeb	q-axis air gap flux		v_str
psi2d	$psi2d$	Algeb	d-axis air gap flux		v_str
psi2	ψ_2	Algeb	air gap flux magnitude		v_str
Se	Se	Algeb	saturation output		v_str
XaqI1q	$XaqI1q$	Algeb	q-axis reaction	*p.u (kV)*	v_str
a	a	ExtAlgeb	Bus voltage phase angle		
v	v	ExtAlgeb	Bus voltage magnitude		

Initialization Equations

Name	Symbol	Type	Initial Value
delta	δ	State	δ_0
omega	ω	State	u
e1q	$e1q$	State	$e1q0u$
e1d	$e1d$	State	$e1d0u$
e2d	$e2d$	State	$e2d0u$
e2q	$e2q$	State	$e2q0u$
Id	Id	Algeb	Id_0u
Iq	Iq	Algeb	Iq_0u
vd	vd	Algeb	uvd_0
vq	vq	Algeb	uvq_0
tm	tm	Algeb	tm_0
te	te	Algeb	tm_0u
vf	vf	Algeb	uvf_0
XadIfd	$XadIfd$	Algeb	uvf_0
Pe	Pe	Algeb	$u\left(Id_0vd_0 + Iq_0vq_0\right)$
Qe	Qe	Algeb	$u\left(Id_0vq_0 - Iq_0vd_0\right)$
psid	$psid$	Algeb	$psid_0u$
psiq	$psiq$	Algeb	$psiq_0u$
psi2q	$psi2q$	Algeb	$psi2q0u$
psi2d	$psi2d$	Algeb	$psi2d0u$
psi2	ψ_2	Algeb	$u\left\|\psi_{20dq}\right\|$
Se	Se	Algeb	Se_0u
XaqI1q	$XaqI1q$	Algeb	0
a	a	ExtAlgeb	
v	v	ExtAlgeb	

Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
delta	δ	State	$2\pi fnue\left(\omega - 1\right)$	
omega	ω	State	$ue\left(-D\left(\omega - 1\right) - te + tm\right)$	M
e1q	$e1q$	State	$-XadIfd + vf$	T'_{d0}
e1d	$e1d$	State	$-XaqI1q$	T'_{q0}
e2d	$e2d$	State	$-Id\left(xd_1 - xl\right) + e1q - e2d$	T''_{d0}
e2q	$e2q$	State	$Iq\left(-xl + xq_1\right) + e1d - e2q$	T''_{q0}

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
Id	Id	Algeb	$Idxd_2 - psi2d + psid$
Iq	Iq	Algeb	$Iqxq_2 + psi2q + psiq$
vd	vd	Algeb	$-uev \sin(a - \delta) - vd$
vq	vq	Algeb	$uev \cos(a - \delta) - vq$
tm	tm	Algeb	$-tm + tm_0$
te	te	Algeb	$-te + ue(-Idpsiq + Iqpsid)$
vf	vf	Algeb	$uevf_0 - vf$
XadIfd	$XadIfd$	Algeb	$-XadIfd + ue(Sepsi2d + e1q + (xd - xd_1)(Idgd_1 + e1qgd_2 - e2dgd_2))$
Pe	Pe	Algeb	$-Pe + ue(Idvd + Iqvq)$
Qe	Qe	Algeb	$-Qe + ue(Idvq - Iqvd)$
psid	$psid$	Algeb	$-psid + ue(Iqra + vq)$
psiq	$psiq$	Algeb	$psiq + ue(Idra + vd)$
psi2q	$psi2q$	Algeb	$e1dgq_1 + e2q(1 - gq_1) - psi2q$
psi2d	$psi2d$	Algeb	$e1qgd_1 + e2dgd_2(xd_1 - xl) - psi2d$
psi2	ψ_2	Algeb	$-\psi_2^2 + psi2d^2 + psi2q^2$
Se	Se	Algeb	$SAT_B SL_{z0}(-SAT_A + \psi_2)^2 - Se\psi_2$
XaqI1q	$XaqI1q$	Algeb	$Segqdpsi2q - XaqI1q + e1d + (xq - xq_1)(-Iqgq_1 + e1dgq_2 - e2qgq_2)$
a	a	ExtAlgeb	$-ue(Idvd + Iqvq)$
v	v	ExtAlgeb	$-ue(Idvq - Iqvd)$

Services

Name	Symbol	Equation	Type
p0	P_0	$gammapp0s$	ConstService
q0	Q_0	$gammaqq0s$	ConstService
gd1	γ_{d1}	$\frac{xd_2 - xl}{xd_1 - xl}$	ConstService
gq1	γ_{q1}	$\frac{-xl + xq_2}{-xl + xq_1}$	ConstService
gd2	γ_{d2}	$\frac{xd_1 - xd_2}{(xd_1 - xl)^2}$	ConstService
gq2	γ_{q2}	$\frac{xq_1 - xq_2}{(-xl + xq_1)^2}$	ConstService
gqd	γ_{qd}	$\frac{-xl + xq}{xd - xl}$	ConstService
_S12	$S_{1.2}$	$S_{12} -_f S12 + 1$	ConstService
SAT_E1	$E_{S_{AT}}^{1c}$	1.0	ConstService
SAT_E2	$E_{S_{AT}}^{2c}$	$3.2 - 2SAT_{zSE2}$	ConstService
SAT_SE1	$SE_{S_{AT}}^{1c}$	S_{10}	ConstService
SAT_SE2	$SE_{S_{AT}}^{2c}$	$S_{12} - 2SAT_{zSE2} + 2$	ConstService

continues on next page

Table 41 – continued from previous page

Name	Symbol	Equation	Type
SAT_a	$a_{S_{AT}}$	$\sqrt{\frac{SAT_{E1}SAT_{SE1}}{SAT_{E2}SAT_{SE2}}}\left(\text{Indicator}\left(SAT_{SE2}>0\right)+\text{Indicator}\left(SAT_{SE2}<0\right)\right)$	ConstService
SAT_A	$A^q_{S_{AT}}$	$SAT_{E2}-\frac{SAT_{E1}-SAT_{E2}}{SAT_a-1}$	ConstService
SAT_B	$B^q_{S_{AT}}$	$\frac{SAT_{E2}SAT_{SE2}(SAT_a-1)^2(\text{Indicator}\left(SAT_a>0\right)+\text{Indicator}\left(SAT_a<0\right))}{\left(SAT_{E1}-SAT_{E2}\right)^2}$	ConstService
_V	V_c	ve^{ia}	ConstService
_S	S	p_0-iq_0	ConstService
_Zs	Z_s	$ra+ixd_2$	ConstService
_It	I_t	$\frac{S}{\text{conj}\left(V\right)}$	ConstService
_Is	I_s	${}_It+\frac{V}{{}_Zs}$	ConstService
psi20	ψ''_0	${}_Is{}_Zs$	ConstService
psi20_arg	$\theta_{\psi''0}$	$\text{arg}\left(\psi_{20}\right)$	ConstService
psi20_abs	$\lvert\psi''_0\rvert$	$\lvert\psi_{20}\rvert$	ConstService
_It_arg	θ_{It0}	$\text{arg}\left({}_It\right)$	ConstService
_psi20_It_arg	$\theta_{\psi aIt}$	$-{}_It_{arg}+\psi_{20arg}$	ConstService
Se0	S_{e0}	$\frac{SAT_B(-SAT_A+\psi_{20abs})^2\,\text{Indicator}\left(\psi_{20abs}\geq SAT_A\right)}{\psi_{20abs}}$	ConstService
_a	a'	$\psi_{20abs}\left(Se_0gqd+1\right)$	ConstService
_b	b'	$\left(-xq+xq_2\right)\lvert{}_It\rvert$	ConstService
delta0	δ_0	$\psi_{20arg}+\text{atan}\left(\frac{{}_b\cos\left({}_psi20_{Itarg}\right)}{-{}_a+{}_b\sin\left({}_psi20_{Itarg}\right)}\right)$	ConstService
_Tdq	T_{dq}	$-i\sin\left(\delta_0\right)+\cos\left(\delta_0\right)$	ConstService
psi20_dq	$\psi''_{0,dq}$	${}_Tdq\psi_{20}$	ConstService
It_dq	$I_{t,dq}$	$\text{conj}\left({}_It{}_Tdq\right)$	ConstService
psi2d0	ψ_{ad0}	$\text{re}\left(\psi_{20dq}\right)$	ConstService
psi2q0	ψ_{aq0}	$-\text{im}\left(\psi_{20dq}\right)$	ConstService
Id0	I_{d0}	$\text{im}\left(It_{dq}\right)$	ConstService
Iq0	I_{q0}	$\text{re}\left(It_{dq}\right)$	ConstService
vd0	V_{d0}	$-Id_0ra+Iq_0xq_2+psi2q0$	ConstService
vq0	V_{q0}	$-Id_0xd_2-Iq_0ra+psi2d0$	ConstService
tm0	τ_{m0}	$u\left(Id_0\left(Id_0ra+vd_0\right)+Iq_0\left(Iq_0ra+vq_0\right)\right)$	ConstService
vf0	v_{f0}	$Id_0\left(xd-xd_2\right)+psi2d0\left(Se_0+1\right)$	ConstService
psid0	ψ_{d0}	$Iq_0rau+vq_0$	ConstService
psiq0	ψ_{q0}	$-Id_0rau-vd_0$	ConstService
e1q0	e'_{q0}	$Id_0\left(-xd+xd_1\right)-Se_0psi2d0+vf_0$	ConstService
e1d0	e'_{d0}	$Iq_0\left(xq-xq_1\right)-Se_0gqdpsi2q0$	ConstService
e2d0	e''_{d0}	$Id_0\left(-xd+xl\right)-Se_0psi2d0+vf_0$	ConstService
e2q0	e''_{q0}	$-Iq_0\left(xl-xq\right)-Se_0gqdpsi2q0$	ConstService

Discretes

Name	Symbol	Type	Info
SL	SL	LessThan	

Blocks

Name	Symbol	Type	Info
SAT	S_{AT}	ExcQuadSat	

Setpoints

Name	Target	Info
pref	tm0	
vref	vf0	

Config Fields in [GENROU]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)
vf_lower		1	lower limit for vf warning	
vf_upper		5	upper limit for vf warning	

PLBVFU1

PLBVFU1 model: playback of voltage and frequency as a generator.

The internal voltage and frequency are named `Vflt` and `omega`. Rotor angle is named `delta`.

The current implementation relies on a `TimeSeries` device to provide the voltage and frequency signals. See `ieee14_plbvfu1.xlsx` and `plbvf.xlsx` in `andes/cases/ieee14` for an example.

Voltage and frequeny data needs to be specified in per unit. Nominal values are not yet supported.

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
bus		interface bus id			mandatory
gen		static generator index			mandatory
Sn	S_n	Power rating	100	*MVA*	
Vn	V_n	AC voltage rating	110		
ra	r_a	armature resistance	0		z
xs	x_s	generator transient reactance	0.200		non_zero,z
fn	f_n	rated frequency	60		
Vflag		playback voltage signal	1	*bool*	
fflag		playback frequency signal	1	*bool*	
filename		playback file name		*string*	mandatory
Vscale	V_{scale}	playback voltage scale	1	*pu*	non_negative
fscale	f_{scale}	playback frequency scale	1	*pu*	non_negative
Tv	T_v	filtering time constant for voltage	0.200	*s*	non_negative
Tf	T_f	filtering time constant for frequency	0.200	*s*	non_negative
ue	u_e	effective online status	0		
subidx		Generator idx in plant; only used by PSS/E data	0		

Variables

Name	Symbol	Type	Description	Unit	Properties
Vflt	$Vflt$	State	filtered voltage	*pu*	v_str
omega	ω	State	filtered frequency	*pu*	v_str
delta	δ	State	rotor angle	*rad*	v_str
a	a	ExtAlgeb	Bus voltage phase angle		
v	v	ExtAlgeb	Bus voltage magnitude		

Initialization Equations

Name	Symbol	Type	Initial Value
Vflt	$Vflt$	State	$-Voffs + VtsiVscale$
omega	ω	State	$-foffs + ftsifscale$
delta	δ	State	δ_0
a	a	ExtAlgeb	
v	v	ExtAlgeb	

Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
Vflt	$Vflt$	State	$-Vflt - Voffs + VtsiVscale$	T_v
omega	ω	State	$-foffs + ftsifscale - \omega$	T_f
delta	δ	State	$2\pi fnue\,(\omega - 1)$	

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
a	a	ExtAlgeb	$\frac{Vfltvxs\sin(a-\delta)}{ra^2+xs^2} + \frac{rav(-Vflt\cos(a-\delta)+v)}{ra^2+xs^2}$
v	v	ExtAlgeb	$-\frac{Vfltrav\sin(a-\delta)}{ra^2+xs^2} + \frac{vxs(-Vflt\cos(a-\delta)+v)}{ra^2+xs^2}$

Services

Name	Symbol	Equation	Type
zs	zs	$ra + ixs$	ConstService
zs2n	$zs2n$	$ra^2 - xs^2$	ConstService
Ec	E_c	$ve^{ia} + (ra + ixs)\,\mathrm{conj}\left(\frac{(p+iq)e^{-ia}}{v}\right)$	ConstService
E0	E_0	$\lvert Ec\rvert$	ConstService
delta0	δ_0	$\arg(Ec)$	ConstService
Vts	Vts	0	ConstService
fts	fts	0	ConstService
ifscale	$1/f_{scale}$	$\frac{1}{fscale}$	ConstService
iVscale	$1/V_{scale}$	$\frac{1}{Vscale}$	ConstService
foffs	f_{offs}	$ftsifscale - 1$	ConstService
Voffs	V_{offs}	$-E_0 + VtsiVscale$	ConstService

Config Fields in [PLBVFU1]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

5.3.35 TimedEvent

Timed event group

Common Parameters: u, name

Available models: *Toggle*, *Fault*, *Alter*

Toggle

Time-based connectivity status toggle.

Toggle is used to toggle the connection status (online/offline) of a device at the predefined time. Both the model name (or group name) and the device idx need to be specified. It effectively negates the `u` field of the connected device.

Toggle can be useful to implement disconnection, connection, and reconnection of devices. For example, a line trip can be implemented by setting `Line` to the `model` field and the corresponding line's `idx` to the `dev` field.

Multiple Toggles can be added to the same device at different times. Adding two Toggles for an initially connected line with `t=0.1` and `t=0.2`, for instance, will disconnect the line at t=0.1 sec and reconnect it at t=0.2 sec.

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
model		model or group name of the device			mandatory
dev		idx of the device to control			mandatory
t		switch time for connection status	-1		mandatory
ue	u_e	effective online status	0		

Services

Name	Symbol	Equation	Type
_u	u	1	ConstService

Config Fields in [Toggle]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

Fault

Three-phase-to-ground fault.

A Fault device is used to apply and clear three-phase-to-ground fault to the given bus. One can set two time parameters, `tf` and `tc`, for the fault-on and fault-clearance time, respectively, although only `tf` is mandatory.

A fault is implemented by a very small internal shunt impedance to be connected at the fault-on time. Its reactance and resistance are specified by the parameters `xf` and `rf`.

To implement a fault and its clearance by tripping a line, one can combine `Fault` and `Toggle`. That is, clear a fault in concurrence with a Toggle. The user needs to ensure data consistency so that the line trip actually clears the fault.

Non-convergence can occur in the proximity of a fault due to various reasons, including network power transfer capability limitation and parameter issues of controllers.

When a fault gets cleared, algebraic variables change drastically. E.g., voltages can go from nearly zero back to 1.0. As we are using Newton's method for solving the DAE, the initial values are crucial for the immediate step after fault clearance.

This Fault model restores the pre-fault values for algebraic variables `Fault.config.scale` is the scaling factor to be multiplied to the pre-fault values for adjusting the initial values. Some trial and error are expected for severe disturbances, combined with increasing the fault reactance `xf`.

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
bus		linked bus idx			mandatory
tf		Bus fault start time	-1	*second*	mandatory
tc		Bus fault end time	-1	*second*	
xf	x_f	Fault to ground reactance (positive)	0.000	*p.u.(sys)*	
rf	x_f	Fault to ground resistance (positive)	0	*p.u.(sys)*	
ue	u_e	effective online status	0		

Variables

Name	Symbol	Type	Description	Unit	Properties
a	a	ExtAlgeb	Bus voltage angle	*p.u.(kV)*	
v	v	ExtAlgeb	Bus voltage magnitude	*p.u.(kV)*	

Initialization Equations

Name	Symbol	Type	Initial Value
a	a	ExtAlgeb	
v	v	ExtAlgeb	

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
a	a	ExtAlgeb	$gfuufv^2$
v	v	ExtAlgeb	$-bfuufv^2$

Services

Name	Symbol	Equation	Type
gf	g_f	$\frac{rf}{rf^2+xf^2}$	ConstService
bf	b_f	$-\frac{xf}{rf^2+xf^2}$	ConstService
uf	u_f	0	ConstService

Config Fields in [Fault]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)
restore		1	restore algebraic variables to pre-fault values	(0, 1)
mode		1	1. restore all algeb variables, 2. fault bus only	(1, 2, 3)
scale		1	scaling factor of restored algebraic values	

Alter

Model for altering device internal data at predefined time.

Alter is useful to apply load changing, tap changing, step response, etc. can be applied to parameters and constant services but cannot be used to update variables.

Alter is implemented by applying the given calculation to the `v` field of the linked parameter or constant. Alter will not affect other parameters or constants that depend on the altered variable.

It is not uncommon for equations to depend on intermediate constants rather than the input parameters. Therefore, one will need to inspect model equations to determine the parameter/service to be altered.

Examples

To apply a PQ load change, according to *PQ*, one needs to set the load model to constant power and alter `Ppf` and `Qpf`. Altering `p0` and `q0` will have no impact as they are not used in the equations for time-domain simulation.

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
t		switch time for connection status	-1		mandatory
model		model or group name of the device			mandatory
dev		idx of the device to alter			mandatory
src		model source field (param or service)			mandatory
attr		attribute (e.g., v) of the source field	v		
method		alteration method in +, -, *, /, =			mandatory
amount		the amount to apply			mandatory
rand		use uniform ramdom sampling	0		
lb		lower bound of random sampling	0		
ub		upper bound of random sampling	0		
ue	u_e	effective online status	0		

Discretes

Name	Symbol	Type	Info
SW	SW	Switcher	Switcher for alteration method

Config Fields in [Alter]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

5.3.36 TurbineGov

Turbine governor group for synchronous generator.

Common Parameters: u, name, syn

Common Variables: pout

Available models: *TG2*, *TGOV1*, *TGOV1DB*, *TGOV1N*, *TGOV1NDB*, *IEEEG1*, *IEESGO*, *GAST*, *HYGOV*, *HYGOVDB*, *HYGOV4*

TG2

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
syn		Synchronous generator idx			mandatory,unique
Tn	T_n	Turbine power rating. Equal to *Sn* if not provided.	0	*MVA*	
wref0	ω_{ref0}	Base speed reference	1	*p.u.*	
R	R	Speed regulation gain (mach. base default)	0.050	*p.u.*	ipower
pmax	p_{max}	Maximum power output	999	*p.u.*	power
pmin	p_{min}	Minimum power output	0	*p.u.*	power
dbl	L_{db}	Deadband lower limit	-0.000	*p.u.*	
dbu	U_{db}	Deadband upper limit	0.000	*p.u.*	
dbc	C_{db}	Deadband neutral value	0	*p.u.*	
T1	T_1	Transient gain time	0.200		
T2	T_2	Governor time constant	10		
ue	u_e	effective online status	0		
Sg	S_n	Rated power from generator	0	*MVA*	
Vn	V_n	Rated voltage from generator	0	*kV*	

Variables

Name	Symbol	Type	Description	Unit	Properties
ll_x	ll_x	State	State in lead-lag		v_str
omega	ω	ExtState	Generator speed	*p.u.*	
paux	$paux$	Algeb	Auxiliary power input		v_str
pout	$pout$	Algeb	Turbine final output power		v_str
wref	$wref$	Algeb	Speed reference variable		v_str
w_d	w_d	Algeb	Generator speed deviation before dead band (positive for under speed)		v_str
w_dm	w_{dm}	Algeb	Measured speed deviation after dead band		v_str
w_dmg	w_{dmg}	Algeb	Speed deviation after dead band after gain		v_str
ll_y	ll_y	Algeb	Output of lead-lag		v_str
pnl	pnl	Algeb	Power output before hard limiter		v_str
tm	tm	ExtAlgeb	Mechanical power interface to SynGen		

Initialization Equations

Name	Symbol	Type	Initial Value
ll_x	ll_x	State	w_{dmg}
omega	ω	ExtState	
paux	$paux$	Algeb	$paux_0$
pout	$pout$	Algeb	tm_0ue
wref	$wref$	Algeb	$wref_0$
w_d	w_d	Algeb	0
w_dm	w_{dm}	Algeb	0
w_dmg	w_{dmg}	Algeb	0
ll_y	ll_y	Algeb	w_{dmg}
pnl	pnl	Algeb	tm_0
tm	tm	ExtAlgeb	

Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
ll_x	ll_x	State	$-ll_x + w_{dmg}$	T_2
omega	ω	ExtState	0	

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
paux	$paux$	Algeb	$-paux + paux_0$
pout	$pout$	Algeb	$plim_{zi}pnl + plim_{zl}pmin + plim_{zu}pmax - pout$
wref	$wref$	Algeb	$-wref + wref_0$
w_d	w_d	Algeb	$ue\left(-\omega + wref\right) - w_d$
w_dm	w_{dm}	Algeb	$dblw_{dbzlr} + dbuw_{dbzur} + w_d\left(1 - w_{dbzi}\right) - w_{dm}$
w_dmg	w_{dmg}	Algeb	$gainw_{dm} - w_{dmg}$
ll_y	ll_y	Algeb	$T_1\left(-ll_x + w_{dmg}\right) + T_2ll_x - T_2ll_y + ll_{LTz1}\left(ll_y - w_{dmg}\right)$
pnl	pnl	Algeb	$ll_y - pnl + pref_0$
tm	tm	ExtAlgeb	$ue\left(pout - tm_0\right)$

Services

Name	Symbol	Equation	Type
pref0	P_{ref0}	tm_0	ConstService
paux0	P_{aux0}	0	ConstService
gain	G	$\frac{u}{R}$	ConstService

Discretes

Name	Symbol	Type	Info
w_db	w_db	DeadBandRT	
ll_LT	LT_{ll}	LessThan	
plim	$plim$	HardLimiter	

Blocks

Name	Symbol	Type	Info
ll	ll	LeadLag	

Setpoints

Name	Target	Info
pref	pref0	initial pref
paux	paux0	const. auxiliary input

Config Fields in [TG2]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)
deadband	$z_{deadband}$	0	enable input dead band	(0, 1)
hardlimit	$z_{hardlimit}$	1	enable output hard limit	(0, 1)

TGOV1

TGOV1 turbine governor model.

Implements the PSS/E TGOV1 model without deadband.

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
syn		Synchronous generator idx			mandatory,unique
Tn	T_n	Turbine power rating. Equal to *Sn* if not provided.	0	*MVA*	
wref0	ω_{ref0}	Base speed reference	1	*p.u.*	
R	R	Speed regulation gain (mach. base default)	0.050	*p.u.*	ipower
VMAX	V_{max}	Maximum valve position	1.200	*p.u.*	power
VMIN	V_{min}	Minimum valve position	0	*p.u.*	power
T1	T_1	Valve time constant	0.100		
T2	T_2	Lead-lag lead time constant	0.200		
T3	T_3	Lead-lag lag time constant	10		
Dt	D_t	Turbine damping coefficient	0		power
ue	u_e	effective online status	0		
Sg	S_n	Rated power from generator	0	*MVA*	
Vn	V_n	Rated voltage from generator	0	*kV*	

Variables

Name	Symbol	Type	Description	Unit	Properties
LAG_y	LAG_y	State	State in lag TF		v_str
LL_x	LL_x	State	State in lead-lag		v_str
omega	ω	ExtState	Generator speed	*p.u.*	
paux	$paux$	Algeb	Auxiliary power input		v_str
pout	$pout$	Algeb	Turbine final output power		v_str
wref	$wref$	Algeb	Speed reference variable		v_str
pref	$pref$	Algeb	Reference power input		v_str
wd	wd	Algeb	Generator speed deviation	*p.u.*	v_str
pd	pd	Algeb	Pref plus speed deviation times gain	*p.u.*	v_str
LL_y	LL_y	Algeb	Output of lead-lag		v_str
tm	tm	ExtAlgeb	Mechanical power interface to SynGen		

Initialization Equations

Name	Symbol	Type	Initial Value
LAG_y	LAG_y	State	pd
LL_x	LL_x	State	LAG_y
omega	ω	ExtState	
paux	$paux$	Algeb	$paux_0$
pout	$pout$	Algeb	tm_0ue
wref	$wref$	Algeb	$wref_0$
pref	$pref$	Algeb	Rtm_0
wd	wd	Algeb	0
pd	pd	Algeb	tm_0ue
LL_y	LL_y	Algeb	LAG_y
tm	tm	ExtAlgeb	

Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
LAG_y	LAG_y	State	$-LAG_y + pd$	T_1
LL_x	LL_x	State	$LAG_y - LL_x$	T_3
omega	ω	ExtState	0	

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
paux	$paux$	Algeb	$-paux + paux_0$
pout	$pout$	Algeb	$-pout + ue\left(-Dtwd + LL_y\right)$
wref	$wref$	Algeb	$-wref + wref_0$
pref	$pref$	Algeb	$Rpref_0 - pref$
wd	wd	Algeb	$ue\left(\omega - wref\right) - wd$
pd	pd	Algeb	$gainue\left(paux + pref - wd\right) - pd$
LL_y	LL_y	Algeb	$LL_{LTz1}\left(-LAG_y + LL_y\right) + LL_xT_3 - LL_yT_3 + T_2\left(LAG_y - LL_x\right)$
tm	tm	ExtAlgeb	$ue\left(pout - tm_0\right)$

Services

Name	Symbol	Equation	Type
pref0	P_{ref0}	tm_0	ConstService
paux0	P_{aux0}	0	ConstService
gain	G	$\frac{ue}{R}$	ConstService

Discretes

Name	Symbol	Type	Info
LAG_lim	lim_{LAG}	AntiWindup	Limiter in Lag
LL_LT	LT_{LL}	LessThan	

Blocks

Name	Symbol	Type	Info
LAG	LAG	LagAntiWindup	
LL	LL	LeadLag	

Setpoints

Name	Target	Info
pref	pref0	initial pref
paux	paux0	const. auxiliary input

Config Fields in [TGOV1]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

TGOV1DB

TGOV1 turbine governor model with speed input deadband.

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
syn		Synchronous generator idx			mandatory,unique
Tn	T_n	Turbine power rating. Equal to *Sn* if not provided.	0	*MVA*	
wref0	ω_{ref0}	Base speed reference	1	*p.u.*	
R	R	Speed regulation gain (mach. base default)	0.050	*p.u.*	ipower
VMAX	V_{max}	Maximum valve position	1.200	*p.u.*	power
VMIN	V_{min}	Minimum valve position	0	*p.u.*	power
T1	T_1	Valve time constant	0.100		
T2	T_2	Lead-lag lead time constant	0.200		
T3	T_3	Lead-lag lag time constant	10		
Dt	D_t	Turbine damping coefficient	0		power
dbL	db_L	Lower bound of deadband	0	*p.u.*	
dbU	db_U	Upper bound of deadband	0	*p.u.*	
ue	u_e	effective online status	0		
Sg	S_n	Rated power from generator	0	*MVA*	
Vn	V_n	Rated voltage from generator	0	*kV*	

Variables

Name	Symbol	Type	Description	Unit	Properties
LAG_y	LAG_y	State	State in lag TF		v_str
LL_x	LL_x	State	State in lead-lag		v_str
omega	ω	ExtState	Generator speed	*p.u.*	
paux	$paux$	Algeb	Auxiliary power input		v_str
pout	$pout$	Algeb	Turbine final output power		v_str
wref	$wref$	Algeb	Speed reference variable		v_str
pref	$pref$	Algeb	Reference power input		v_str
wd	wd	Algeb	Generator speed deviation	*p.u.*	v_str
pd	pd	Algeb	Pref plus speed deviation times gain	*p.u.*	v_str
LL_y	LL_y	Algeb	Output of lead-lag		v_str
DB_y	DB_y	Algeb	Deadband type 1 output		v_str
tm	tm	ExtAlgeb	Mechanical power interface to SynGen		

Initialization Equations

Name	Symbol	Type	Initial Value
LAG_y	LAG_y	State	pd
LL_x	LL_x	State	LAG_y
omega	ω	ExtState	
paux	$paux$	Algeb	$paux_0$
pout	$pout$	Algeb	tm_0ue
wref	$wref$	Algeb	$wref_0$
pref	$pref$	Algeb	Rtm_0
wd	wd	Algeb	0
pd	pd	Algeb	tm_0ue
LL_y	LL_y	Algeb	LAG_y
DB_y	DB_y	Algeb	$1.0DB_{dbzl}\left(-dbL + wd\right) + 1.0DB_{dbzu}\left(-dbU + wd\right)$
tm	tm	ExtAlgeb	

Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
LAG_y	LAG_y	State	$-LAG_y + pd$	T_1
LL_x	LL_x	State	$LAG_y - LL_x$	T_3
omega	ω	ExtState	0	

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
paux	$paux$	Algeb	$-paux + paux_0$
pout	$pout$	Algeb	$-DB_y Dt + LL_y - pout$
wref	$wref$	Algeb	$-wref + wref_0$
pref	$pref$	Algeb	$Rpref_0 - pref$
wd	wd	Algeb	$ue\left(\omega - wref\right) - wd$
pd	pd	Algeb	$gainue\left(-DB_y + paux + pref\right) - pd$
LL_y	LL_y	Algeb	$LL_{LTz1}\left(-LAG_y + LL_y\right) + LL_x T_3 - LL_y T_3 + T_2\left(LAG_y - LL_x\right)$
DB_y	DB_y	Algeb	$1.0DB_{dbzl}\left(-dbL + wd\right) + 1.0DB_{dbzu}\left(-dbU + wd\right) - DB_y$
tm	tm	ExtAlgeb	$ue\left(pout - tm_0\right)$

Services

Name	Symbol	Equation	Type
pref0	P_{ref0}	tm_0	ConstService
paux0	P_{aux0}	0	ConstService
gain	G	$\frac{ue}{R}$	ConstService

Discretes

Name	Symbol	Type	Info
LAG_lim	lim_{LAG}	AntiWindup	Limiter in Lag
LL_LT	LT_{LL}	LessThan	
DB_db	db_{DB}	DeadBand	

Blocks

Name	Symbol	Type	Info
LAG	LAG	LagAntiWindup	
LL	LL	LeadLag	
DB	DB	DeadBand1	deadband for speed deviation

Setpoints

Name	Target	Info
pref	pref0	initial pref
paux	paux0	const. auxiliary input

Config Fields in [TGOV1DB]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

TGOV1N

New TGOV1 (TGOV1N) turbine governor model.

The TGOV1N model sums `pref` and `paux` signals after the droop. This model is useful for incorporating AGC and scheduling signals, which will not be multiplied by `1/R` as in the original TGOV1 model.

The recommended approach for modifying setpoints is the group-level API, which resolves the controller chain automatically:

```
ss.SynGen.set_pref(ss, gen_idx, value)   # writes to pref0
ss.SynGen.set_paux(ss, gen_idx, value)    # writes to paux0
```

Modifying `tm0` is not allowed.

For bulk operations, direct in-place array assignment can be used:

```
ss.TGOV1N.paux0.v[:] = paux_new  # in-place update of the paux0.v array
```

The array reference must not be replaced:

```
ss.TGOV1N.paux0.v = paux_new  # error; disconnects the variable
```

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
syn		Synchronous generator idx			mandatory,unique
Tn	T_n	Turbine power rating. Equal to *Sn* if not provided.	0	*MVA*	
wref0	ω_{ref0}	Base speed reference	1	*p.u.*	
R	R	Speed regulation gain (mach. base default)	0.050	*p.u.*	ipower
VMAX	V_{max}	Maximum valve position	1.200	*p.u.*	power
VMIN	V_{min}	Minimum valve position	0	*p.u.*	power
T1	T_1	Valve time constant	0.100		
T2	T_2	Lead-lag lead time constant	0.200		
T3	T_3	Lead-lag lag time constant	10		
Dt	D_t	Turbine damping coefficient	0		power
ue	u_e	effective online status	0		
Sg	S_n	Rated power from generator	0	*MVA*	
Vn	V_n	Rated voltage from generator	0	*kV*	

Variables

Name	Symbol	Type	Description	Unit	Properties
LAG_y	LAG_y	State	State in lag TF		v_str
LL_x	LL_x	State	State in lead-lag		v_str
omega	ω	ExtState	Generator speed	*p.u.*	
paux	$paux$	Algeb	Auxiliary power input		v_str
pout	$pout$	Algeb	Turbine final output power		v_str
wref	$wref$	Algeb	Speed reference variable		v_str
pref	$pref$	Algeb	Reference power input		v_str
wd	wd	Algeb	Generator speed deviation	*p.u.*	v_str
pd	pd	Algeb	Pref plus speed deviation times gain	*p.u.*	v_str
LL_y	LL_y	Algeb	Output of lead-lag		v_str
tm	tm	ExtAlgeb	Mechanical power interface to SynGen		

Initialization Equations

Name	Symbol	Type	Initial Value
LAG_y	LAG_y	State	pd
LL_x	LL_x	State	LAG_y
omega	ω	ExtState	
paux	$paux$	Algeb	$paux_0$
pout	$pout$	Algeb	tm_0ue
wref	$wref$	Algeb	$wref_0$
pref	$pref$	Algeb	tm_0
wd	wd	Algeb	0
pd	pd	Algeb	tm_0ue
LL_y	LL_y	Algeb	LAG_y
tm	tm	ExtAlgeb	

Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
LAG_y	LAG_y	State	$-LAG_y + pd$	T_1
LL_x	LL_x	State	$LAG_y - LL_x$	T_3
omega	ω	ExtState	0	

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
paux	$paux$	Algeb	$-paux + paux_0$
pout	$pout$	Algeb	$-pout + ue\left(-Dtwd + LL_y\right)$
wref	$wref$	Algeb	$-wref + wref_0$
pref	$pref$	Algeb	$-pref + pref_0$
wd	wd	Algeb	$ue\left(\omega - wref\right) - wd$
pd	pd	Algeb	$-pd + ue\left(-gainwd + paux + pref\right)$
LL_y	LL_y	Algeb	$LL_{LTz1}\left(-LAG_y + LL_y\right) + LL_xT_3 - LL_yT_3 + T_2\left(LAG_y - LL_x\right)$
tm	tm	ExtAlgeb	$ue\left(pout - tm_0\right)$

Services

Name	Symbol	Equation	Type
pref0	P_{ref0}	tm_0	ConstService
paux0	P_{aux0}	0	ConstService
gain	G	$\frac{ue}{R}$	ConstService

Discretes

Name	Symbol	Type	Info
LAG_lim	lim_{LAG}	AntiWindup	Limiter in Lag
LL_LT	LT_{LL}	LessThan	

Blocks

Name	Symbol	Type	Info
LAG	LAG	LagAntiWindup	
LL	LL	LeadLag	

Setpoints

Name	Target	Info
pref	pref0	initial pref
paux	paux0	const. auxiliary input

Config Fields in [TGOV1N]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

TGOV1NDB

TGOV1N turbine governor model with speed input deadband.

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
syn		Synchronous generator idx			mandatory,unique
Tn	T_n	Turbine power rating. Equal to *Sn* if not provided.	0	*MVA*	
wref0	ω_{ref0}	Base speed reference	1	*p.u.*	
R	R	Speed regulation gain (mach. base default)	0.050	*p.u.*	ipower
VMAX	V_{max}	Maximum valve position	1.200	*p.u.*	power
VMIN	V_{min}	Minimum valve position	0	*p.u.*	power
T1	T_1	Valve time constant	0.100		
T2	T_2	Lead-lag lead time constant	0.200		
T3	T_3	Lead-lag lag time constant	10		
Dt	D_t	Turbine damping coefficient	0		power
dbL	db_L	Lower bound of deadband	0	*p.u.*	
dbU	db_U	Upper bound of deadband	0	*p.u.*	
ue	u_e	effective online status	0		
Sg	S_n	Rated power from generator	0	*MVA*	
Vn	V_n	Rated voltage from generator	0	*kV*	

Variables

Name	Symbol	Type	Description	Unit	Properties
LAG_y	LAG_y	State	State in lag TF		v_str
LL_x	LL_x	State	State in lead-lag		v_str
omega	ω	ExtState	Generator speed	*p.u.*	
paux	$paux$	Algeb	Auxiliary power input		v_str
pout	$pout$	Algeb	Turbine final output power		v_str
wref	$wref$	Algeb	Speed reference variable		v_str
pref	$pref$	Algeb	Reference power input		v_str
wd	wd	Algeb	Generator speed deviation	*p.u.*	v_str
pd	pd	Algeb	Pref plus speed deviation times gain	*p.u.*	v_str
LL_y	LL_y	Algeb	Output of lead-lag		v_str
DB_y	DB_y	Algeb	Deadband type 1 output		v_str
tm	tm	ExtAlgeb	Mechanical power interface to SynGen		

Initialization Equations

Name	Symbol	Type	Initial Value
LAG_y	LAG_y	State	pd
LL_x	LL_x	State	LAG_y
omega	ω	ExtState	
paux	$paux$	Algeb	$paux_0$
pout	$pout$	Algeb	$tm_0 ue$
wref	$wref$	Algeb	$wref_0$
pref	$pref$	Algeb	tm_0
wd	wd	Algeb	0
pd	pd	Algeb	$tm_0 ue$
LL_y	LL_y	Algeb	LAG_y
DB_y	DB_y	Algeb	$1.0DB_{dbzl}\left(-dbL + wd\right) + 1.0DB_{dbzu}\left(-dbU + wd\right)$
tm	tm	ExtAlgeb	

Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
LAG_y	LAG_y	State	$-LAG_y + pd$	T_1
LL_x	LL_x	State	$LAG_y - LL_x$	T_3
omega	ω	ExtState	0	

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
paux	$paux$	Algeb	$-paux + paux_0$
pout	$pout$	Algeb	$-DB_y Dt + LL_y - pout$
wref	$wref$	Algeb	$-wref + wref_0$
pref	$pref$	Algeb	$-pref + pref_0$
wd	wd	Algeb	$ue\left(\omega - wref\right) - wd$
pd	pd	Algeb	$-pd + ue\left(-DB_y gain + paux + pref\right)$
LL_y	LL_y	Algeb	$LL_{LTz1}\left(-LAG_y + LL_y\right) + LL_x T_3 - LL_y T_3 + T_2\left(LAG_y - LL_x\right)$
DB_y	DB_y	Algeb	$1.0DB_{dbzl}\left(-dbL + wd\right) + 1.0DB_{dbzu}\left(-dbU + wd\right) - DB_y$
tm	tm	ExtAlgeb	$ue\left(pout - tm_0\right)$

Services

Name	Symbol	Equation	Type
pref0	P_{ref0}	tm_0	ConstService
paux0	P_{aux0}	0	ConstService
gain	G	$\frac{ue}{R}$	ConstService

Discretes

Name	Symbol	Type	Info
LAG_lim	lim_{LAG}	AntiWindup	Limiter in Lag
LL_LT	LT_{LL}	LessThan	
DB_db	db_{DB}	DeadBand	

Blocks

Name	Symbol	Type	Info
LAG	LAG	LagAntiWindup	
LL	LL	LeadLag	
DB	DB	DeadBand1	deadband for speed deviation

Setpoints

Name	Target	Info
pref	pref0	initial pref
paux	paux0	const. auxiliary input

Config Fields in [TGOV1NDB]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

IEEEG1

IEEE Type 1 Speed-Governing Model.

If only one generator is connected, its *idx* must be given to *syn*, and *syn2* must be left blank. Each generator must provide data in its *Sn* base.

syn is connected to the high-pressure output (PHP) and the optional *syn2* is connected to the low- pressure output (PLP).

The speed deviation of generator 1 (syn) is measured. If the turbine rating *Tn* is not specified, the sum of *Sn* of all connected generators will be used.

Normally, K1 + K2 + ... + K8 = 1.0. If the second generator is not connected, K1 + K3 + K5 + K7 = 1, and K2 + K4 + K6 + K8 = 0. If K1 to K8 do not sum up to 1.0, they will be normalized. The normalized parameters are called `K1n` through `K8n`.

If initialization error occurs for variable `vs`, it is due to the limits `PMIN` and `PMAX`.

IEEEG1 does not yet support the change of reference (scheduling).

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
syn		Synchronous generator idx			mandatory,unique
Tn	T_n	Turbine power rating. Equal to *Sn* if not provided.	0	*MVA*	
wref0	ω_{ref0}	Base speed reference	1	*p.u.*	

continues on next page

Table 42 – continued from previous page

Name	Symbol	Description	Default	Unit	Properties
syn2		Optional SynGen idx			
K	K	Gain (1/R) in mach. base	20	*p.u. (power)*	power
T1	T_1	Gov. lag time const.	1		
T2	T_2	Gov. lead time const.	1		
T3	T_3	Valve controller time const.	0.100		
UO	U_o	Max. valve opening rate	0.100	*p.u./sec*	
UC	U_c	Max. valve closing rate	-0.100	*p.u./sec*	
PMAX	P_{MAX}	Max. turbine power	5		power
PMIN	P_{MIN}	Min. turbine power	0		power
T4	T_4	Inlet piping/steam bowl time constant	0.400		
K1	K_1	Fraction of power from HP	0.500		
K2	K_2	Fraction of power from LP	0		
T5	T_5	Time constant of 2nd boiler pass	8		
K3	K_3	Fraction of HP shaft power after 2nd boiler pass	0.500		
K4	K_4	Fraction of LP shaft power after 2nd boiler pass	0		
T6	T_6	Time constant of 3rd boiler pass	0.500		
K5	K_5	Fraction of HP shaft power after 3rd boiler pass	0		
K6	K_6	Fraction of LP shaft power after 3rd boiler pass	0		
T7	T_7	Time constant of 4th boiler pass	0.050		
K7	K_7	Fraction of HP shaft power after 4th boiler pass	0		
K8	K_8	Fraction of LP shaft power after 4th boiler pass	0		
ue	u_e	effective online status	0		
Sg	S_n	Rated power from generator	0	*MVA*	
Vn	V_n	Rated voltage from generator	0	*kV*	
Sg2	S_{n2}	Rated power of Syn2	0	*MVA*	

Variables

Name	Symbol	Type	Description	Unit	Properties
LL_x	LL_x	State	State in lead-lag		v_str
IAW_y	IAW_y	State	AW Integrator output		v_str
L4_y	L_{4y}	State	State in lag transfer function		v_str
L5_y	L_{5y}	State	State in lag transfer function		v_str
L6_y	L_{6y}	State	State in lag transfer function		v_str
L7_y	L_{7y}	State	State in lag transfer function		v_str
omega	ω	ExtState	Generator speed	*p.u.*	
paux	$paux$	Algeb	Auxiliary power input		v_str
pout	$pout$	Algeb	Turbine final output power		v_str
wref	$wref$	Algeb	Speed reference variable		v_str
wd	wd	Algeb	Generator under speed	*p.u.*	v_str
LL_y	LL_y	Algeb	Output of lead-lag		v_str
vs	vs	Algeb	Valve speed		v_str
vsl	vsl	Algeb	Valve move speed after limiter		v_str
PHP	PHP	Algeb	HP output		v_str
PLP	PLP	Algeb	LP output		v_str
tm	tm	ExtAlgeb	Mechanical power interface to SynGen		
tm2	tm_2	ExtAlgeb	Mechanical power to syn2		

Initialization Equations

Name	Symbol	Type	Initial Value
LL_x	LL_x	State	wd
IAW_y	IAW_y	State	tm_{012}
L4_y	L_{4y}	State	IAW_y
L5_y	L_{5y}	State	L_{4y}
L6_y	L_{6y}	State	L_{5y}
L7_y	L_{7y}	State	L_{6y}
omega	ω	ExtState	
paux	$paux$	Algeb	$paux_0$
pout	$pout$	Algeb	$tm_0 ue$
wref	$wref$	Algeb	$wref_0$
wd	wd	Algeb	0
LL_y	LL_y	Algeb	wd
vs	vs	Algeb	0
vsl	vsl	Algeb	$HL_{zi}vs + HL_{zl}UC + HL_{zu}UO$
PHP	PHP	Algeb	$ue\left(K1nL_{4y} + K3nL_{5y} + K5nL_{6y} + K7nL_{7y}\right)$
PLP	PLP	Algeb	$ue\left(K2nL_{4y} + K4nL_{5y} + K6nL_{6y} + K8nL_{7y}\right)$
tm	tm	ExtAlgeb	
tm2	tm_2	ExtAlgeb	

Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
LL_x	LL_x	State	$-LL_x + wd$	T_1
IAW_y	IAW_y	State	vsl	1
L4_y	L_{4y}	State	$IAW_y - L_{4y}$	T_4
L5_y	L_{5y}	State	$L_{4y} - L_{5y}$	T_5
L6_y	L_{6y}	State	$L_{5y} - L_{6y}$	T_6
L7_y	L_{7y}	State	$L_{6y} - L_{7y}$	T_7
omega	ω	ExtState	0	

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
paux	$paux$	Algeb	$-paux + paux_0$
pout	$pout$	Algeb	$PHPue - pout$
wref	$wref$	Algeb	$-wref + wref_0$
wd	wd	Algeb	$ue\left(-\omega + wref\right) - wd$
LL_y	LL_y	Algeb	$KLL_xT_1 + KT_2\left(-LL_x + wd\right) + LL_{LTz1}\left(-Kwd + LL_y\right) - LL_yT_1$
vs	vs	Algeb	$-vs + \frac{ue(-IAW_y + LL_y + paux + tm_{012})}{T_3}$
vsl	vsl	Algeb	$HL_{zi}vs + HL_{zl}UC + HL_{zu}UO - vsl$
PHP	PHP	Algeb	$-PHP + ue\left(K1nL_{4y} + K3nL_{5y} + K5nL_{6y} + K7nL_{7y}\right)$
PLP	PLP	Algeb	$-PLP + ue\left(K2nL_{4y} + K4nL_{5y} + K6nL_{6y} + K8nL_{7y}\right)$
tm	tm	ExtAlgeb	$ue\left(pout - tm_0\right)$
tm2	tm_2	ExtAlgeb	$uezsyn_2\left(PLP - tm_{02}\right)$

Services

Name	Symbol	Equation	Type
pref0	P_{ref0}	tm_0	ConstService
paux0	P_{aux0}	0	ConstService
sumK18	$\sum_{i=1}^{8} K_i$	$K_1 + K_2 + K_3 + K_4 + K_5 + K_6 + K_7 + K_8$	ConstService
Kcoeff	K_{coeff}	$\frac{1}{sumK_{18}}$	ConstService
K1n	K_{1n}	$K_1 Kcoeff$	ConstService
K2n	K_{2n}	$K_2 Kcoeff$	ConstService
K3n	K_{3n}	$K_3 Kcoeff$	ConstService
K4n	K_{4n}	$K_4 Kcoeff$	ConstService
K5n	K_{5n}	$K_5 Kcoeff$	ConstService
K6n	K_{6n}	$K_6 Kcoeff$	ConstService
K7n	K_{7n}	$K_7 Kcoeff$	ConstService
K8n	K_{8n}	$K_8 Kcoeff$	ConstService
_tm0K2	${}_t m0K2$	$tm_0 zsyn_2 \left(K2n + K4n + K6n + K8n\right)$	PostInitService
_tm02K1	${}_t m02K1$	$tm_{02} \left(K1n + K3n + K5n + K7n\right)$	PostInitService
tm012	$tm012$	$tm_0 + tm_{02}$	ConstService

Discretes

Name	Symbol	Type	Info
LL_LT	LT_{LL}	LessThan	
HL	HL	HardLimiter	Limiter on valve speed
IAW_lim	lim_{IAW}	AntiWindup	Limiter in integrator

Blocks

Name	Symbol	Type	Info
LL	LL	LeadLag	Signal conditioning for wd
IAW	IAW	IntegratorAntiWindup	Valve position integrator
L4	$L4$	Lag	first process
L5	$L5$	Lag	second (reheat) process
L6	$L6$	Lag	third process
L7	$L7$	Lag	fourth (second reheat) process

Setpoints

Name	Target	Info
pref	pref0	initial pref
paux	paux0	const. auxiliary input

Config Fields in [IEEEG1]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

IEESGO

IEEE Standard Governor (IEESGO).

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
syn		Synchronous generator idx			mandatory,unique
Tn	T_n	Turbine power rating. Equal to *Sn* if not provided.	0	*MVA*	
wref0	ω_{ref0}	Base speed reference	1	*p.u.*	
T1	T_1	Controller lag	0.020		
T2	T_2	Lead compensation	1		
T3	T_3	Governor lag	1		
T4	T_4	Steam inlet delay	0.500		
T5	T_5	Reheater delay	10		
T6	T_6	Crossover delay	0.500		
K1	K_1	1/pu regulation	0.020		
K2	K_2	fraction K2	1		
K3	K_3	fraction K3	1		
PMAX	P_{MAX}	Max. turbine power	5		power
PMIN	P_{MIN}	Min. turbine power	0		power
ue	u_e	effective online status	0		
Sg	S_n	Rated power from generator	0	*MVA*	
Vn	V_n	Rated voltage from generator	0	*kV*	

Variables

Name	Symbol	Type	Description	Unit	Properties
F1_y	F_{1y}	State	State in lag transfer function		v_str
F2_x	F_{2x}	State	State in lead-lag		v_str
F3_y	F_{3y}	State	State in lag transfer function		v_str
F4_y	F_{4y}	State	State in lag transfer function		v_str
F5_y	F_{5y}	State	State in lag transfer function		v_str
omega	ω	ExtState	Generator speed	*p.u.*	
paux	$paux$	Algeb	Auxiliary power input		v_str
pout	$pout$	Algeb	Turbine final output power		v_str
wref	$wref$	Algeb	Speed reference variable		v_str
F2_y	F_{2y}	Algeb	Output of lead-lag		v_str
HL_x	HL_x	Algeb	Value before limiter		v_str
tm	tm	ExtAlgeb	Mechanical power interface to SynGen		

Initialization Equations

Name	Symbol	Type	Initial Value
F1_y	F_{1y}	State	$K_1 ue\left(\omega - wref\right)$
F2_x	F_{2x}	State	F_{1y}
F3_y	F_{3y}	State	$1.0HL_{limzi}HL_x + 1.0HL_{limzl}PMIN + 1.0HL_{limzu}PMAX$
F4_y	F_{4y}	State	$F_{3y}K_2$
F5_y	F_{5y}	State	$F_{4y}K_3$
omega	ω	ExtState	
paux	$paux$	Algeb	$paux_0$
pout	$pout$	Algeb	$tm_0 ue$
wref	$wref$	Algeb	$wref_0$
F2_y	F_{2y}	Algeb	F_{1y}
HL_x	HL_x	Algeb	$1.0ue\left(-F_{2y} + paux + pref_0\right)$
tm	tm	ExtAlgeb	

Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
F1_y	F_{1y}	State	$-F_{1y} + K_1 ue\left(\omega - wref\right)$	T_1
F2_x	F_{2x}	State	$F_{1y} - F_{2x}$	T_3
F3_y	F_{3y}	State	$-F_{3y} + 1.0HL_{limzi}HL_x + 1.0HL_{limzl}PMIN + 1.0HL_{limzu}PMAX$	T_4
F4_y	F_{4y}	State	$F_{3y}K_2 - F_{4y}$	T_5
F5_y	F_{5y}	State	$F_{4y}K_3 - F_{5y}$	T_6
omega	ω	ExtState	0	

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
paux	$paux$	Algeb	$-paux + paux_0$
pout	$pout$	Algeb	$-pout + ue\left(F_{3y}\left(1 - K_2\right) + F_{4y}\left(1 - K_3\right) + F_{5y}\right)$
wref	$wref$	Algeb	$-wref + wref_0$
F2_y	F_{2y}	Algeb	$F_{2LTz1}\left(-1.0F_{1y} + F_{2y}\right) + 1.0F_{2x}T_3 - F_{2y}T_3 + 1.0T_2\left(F_{1y} - F_{2x}\right)$
HL_x	HL_x	Algeb	$-HL_x + 1.0ue\left(-F_{2y} + paux + pref_0\right)$
tm	tm	ExtAlgeb	$ue\left(pout - tm_0\right)$

Services

Name	Symbol	Equation	Type
pref0	P_{ref0}	tm_0	ConstService
paux0	P_{aux0}	0	ConstService

Discretes

Name	Symbol	Type	Info
F2_LT	LT_{F2}	LessThan	
HL_lim	lim_{HL}	HardLimiter	

Blocks

Name	Symbol	Type	Info
F1	$F1$	Lag	
F2	$F2$	LeadLag	
HL	HL	GainLimiter	
F3	$F3$	Lag	
F4	$F4$	Lag	
F5	$F5$	Lag	

Setpoints

Name	Target	Info
pref	pref0	initial pref
paux	paux0	const. auxiliary input

Config Fields in [IEESGO]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

GAST

GAST turbine governor model.

Reference:

[1] Neplan, TURBINE-GOVERNOR GAST, [Online],

Available:

https://www.neplan.ch/wp-content/uploads/2015/08/Nep_TURBINES_GOV.pdf

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
syn		Synchronous generator idx			mandatory,unique
Tn	T_n	Turbine power rating. Equal to *Sn* if not provided.	0	*MVA*	
wref0	ω_{ref0}	Base speed reference	1	*p.u.*	
R	R	Speed regulation gain (mach. base default)	0.050	*p.u.*	ipower
VMAX	V_{max}	Maximum valve position	1.200	*p.u.*	power
VMIN	V_{min}	Minimum valve position	0	*p.u.*	power
KT	K_T	Temperature limiter gain	5		
AT	A_T	Ambient temperature load limit	1		power
T1	T_1	Valve time constant	0.100		
T2	T_2	Lead-lag lead time constant	0.200		
T3	T_3	Lead-lag lag time constant	10		
Dt	D_t	Turbine damping coefficient	0		power
ue	u_e	effective online status	0		
Sg	S_n	Rated power from generator	0	*MVA*	
Vn	V_n	Rated voltage from generator	0	*kV*	

Variables

Name	Symbol	Type	Description	Unit	Properties
LAG_y	LAG_y	State	State in lag TF		v_str
LG2_y	LG_{2y}	State	State in lag transfer function		v_str
LG3_y	LG_{3y}	State	State in lag transfer function		v_str
omega	ω	ExtState	Generator speed	*p.u.*	
paux	$paux$	Algeb	Auxiliary power input		v_str
pout	$pout$	Algeb	Turbine final output power		v_str
wref	$wref$	Algeb	Speed reference variable		v_str
pref	$pref$	Algeb	Reference power input		v_str
wd	wd	Algeb	Generator under speed	*p.u.*	v_str
pd	pd	Algeb	Pref plus under speed times gain	*p.u.*	v_str
v9	v_9	Algeb	V_9 for LVGate input		v_str
tm	tm	ExtAlgeb	Mechanical power interface to SynGen		

Initialization Equations

Name	Symbol	Type	Initial Value
LAG_y	LAG_y	State	$LVG_{ltz0}v_9 + LVG_{ltz1}pd$
LG2_y	LG_{2y}	State	LAG_y
LG3_y	LG_{3y}	State	LG_{2y}
omega	ω	ExtState	
paux	$paux$	Algeb	$paux_0$
pout	$pout$	Algeb	tm_0ue
wref	$wref$	Algeb	$wref_0$
pref	$pref$	Algeb	Rtm_0
wd	wd	Algeb	0
pd	pd	Algeb	tm_0ue
v9	v_9	Algeb	$ue\left(AT + KT\left(AT - tm_0\right)\right)$
tm	tm	ExtAlgeb	

Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
LAG_y	LAG_y	State	$-LAG_y + LVG_{ltz0}v_9 + LVG_{ltz1}pd$	T_1
LG2_y	LG_{2y}	State	$LAG_y - LG_{2y}$	T_2
LG3_y	LG_{3y}	State	$LG_{2y} - LG_{3y}$	T_3
omega	ω	ExtState	0	

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
paux	$paux$	Algeb	$-paux + paux_0$
pout	$pout$	Algeb	$-pout + ue\left(-Dtwd + LG_{2y}\right)$
wref	$wref$	Algeb	$-wref + wref_0$
pref	$pref$	Algeb	$Rpref_0 - pref$
wd	wd	Algeb	$ue\left(\omega - wref\right) - wd$
pd	pd	Algeb	$gainue\left(paux + pref - wd\right) - pd$
v9	v_9	Algeb	$ue\left(AT + KT\left(AT - LG_{3y}\right)\right) - v_9$
tm	tm	ExtAlgeb	$ue\left(pout - tm_0\right)$

Services

Name	Symbol	Equation	Type
pref0	P_{ref0}	tm_0	ConstService
paux0	P_{aux0}	0	ConstService
gain	G	$\frac{ue}{R}$	ConstService

Discretes

Name	Symbol	Type	Info
LVG_lt	$None_{LVG}$	LessThan	
LAG_lim	lim_{LAG}	AntiWindup	Limiter in Lag

Blocks

Name	Symbol	Type	Info
LVG	LVG	LVGate	LVGate
LAG	LAG	LagAntiWindup	
LG2	$LG2$	Lag	Lag T2
LG3	$LG3$	Lag	Lag T3

Setpoints

Name	Target	Info
pref	pref0	initial pref
paux	paux0	const. auxiliary input

Config Fields in [GAST]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

HYGOV

HYGOV turbine governor model.

Implements the PSS/E HYGOV model without deadband.

Reference:

[1] PSSE, Model Library, HYGOV

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
syn		Synchronous generator idx			mandatory,unique
Tn	T_n	Turbine power rating. Equal to *Sn* if not provided.	0	*MVA*	
wref0	ω_{ref0}	Base speed reference	1	*p.u.*	
R	R	Speed regulation gain (mach. base default)	0.050	*p.u.*	ipower
r	r	Temporary droop (R<r)	1	*p.u.*	ipower
GMAX	G_{max}	Maximum governor response	1	*p.u.*	power
GMIN	G_{min}	Minimum governor response	0	*p.u.*	power
VELM	$VELM$	Gate velocity limit	0.300	*p.u.*	power
Tf	T_f	Filter time constant	0.050		
Tr	T_r	Governor time constant	1		
Tg	T_g	Servo time constant	0.050		
Dt	D_t	Turbine damping coefficient	0		power
qNL	q_{NL}	No-load flow at nominal head	0.100		power
Tw	T_w	Water inertia time constant constant	1		
At	A_t	Turbine gain	1		
ue	u_e	effective online status	0		
Sg	S_n	Rated power from generator	0	*MVA*	
Vn	V_n	Rated voltage from generator	0	*kV*	

Variables

Name	Symbol	Type	Description	Unit	Properties
LG_y	LG_y	State	State in lag transfer function		v_str
gtpos	$gtpos$	State	State in gate position (c)	*rad*	v_str
LAG_y	LAG_y	State	State in lag transfer function		v_str
q_y	q_y	State	Integrator output		v_str
omega	ω	ExtState	Generator speed	*p.u.*	
paux	$paux$	Algeb	Auxiliary power input		v_str
pout	$pout$	Algeb	Turbine final output power		v_str
wref	$wref$	Algeb	Speed reference variable		v_str
pref	$pref$	Algeb	Reference power input		v_str
wd	wd	Algeb	Generator speed deviation	*p.u.*	v_str
pd	pd	Algeb	Pref plus speed deviation times gain	*p.u.*	v_str
dg	dg	Algeb	desired gate (c)	*p.u.*	v_str
h	h	Algeb	turbine head	*p.u.*	v_str
tm	tm	ExtAlgeb	Mechanical power interface to SynGen		

Initialization Equations

Name	Symbol	Type	Initial Value
LG_y	LG_y	State	pd
gtpos	$gtpos$	State	q_0
LAG_y	LAG_y	State	dg
q_y	q_y	State	q_0
omega	ω	ExtState	
paux	$paux$	Algeb	$paux_0$
pout	$pout$	Algeb	tm_0ue
wref	$wref$	Algeb	$wref_0$
pref	$pref$	Algeb	Rq_0
wd	wd	Algeb	0
pd	pd	Algeb	0
dg	dg	Algeb	q_0
h	h	Algeb	1
tm	tm	ExtAlgeb	

Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
LG_y	LG_y	State	$-LG_y + pd$	T_f
gtpos	$gtpos$	State	LG_y	
LAG_y	LAG_y	State	$-LAG_y + dg$	T_g
q_y	q_y	State	$1 - \frac{q_y^2}{LAG_y^2}$	T_w
omega	ω	ExtState	0	

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
paux	$paux$	Algeb	$-paux + paux_0$
pout	$pout$	Algeb	$-pout + ue\left(Ath\left(-qNL + q_y\right) - DtLAG_y wd\right)$
wref	$wref$	Algeb	$-wref + wref_0$
pref	$pref$	Algeb	$Rq_0 - pref$
wd	wd	Algeb	$ue\left(\omega - wref\right) - wd$
pd	pd	Algeb	$-pd + ue\left(-Rdg + paux + pref - wd\right)$
dg	dg	Algeb	$LG_y gr - dg + gtpos$
h	h	Algeb	$-h + \frac{q_y^2}{LAG_y^2}$
tm	tm	ExtAlgeb	$ue\left(pout - tm_0\right)$

Services

Name	Symbol	Equation	Type
pref0	P_{ref0}	tm_0	ConstService
paux0	P_{aux0}	0	ConstService
VELMn	$-VELM$	$-VELM$	ConstService
tr	$r * Tr$	Trr	ConstService
gr	$1/r$	$\frac{1}{r}$	ConstService
ratel	$rate_l$	$-VELM - gr$	ConstService
rateu	$rate_u$	$VELM - gr$	ConstService
q0	q_0	$qNL + \frac{tm_0}{At}$	ConstService
dgl	dg_{lower}	$-LG_y gr - VELM$	VarService
dgu	dg_{upper}	$-LG_y gr + VELM$	VarService

Discretes

Name	Symbol	Type	Info
dg_lim	lim_{dg}	AntiWindupRate	gate velocity and position limiter

Blocks

Name	Symbol	Type	Info
LG	LG	Lag	filter after speed deviation (e)
LAG	LAG	Lag	gate opening (g)
q	q	Integrator	turbine flow (q)

Setpoints

Name	Target	Info
pref	pref0	initial pref
paux	paux0	const. auxiliary input

Config Fields in [HYGOV]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

HYGOVDB

HYGOV turbine governor model with speed input deadband.

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
syn		Synchronous generator idx			mandatory,unique
Tn	T_n	Turbine power rating. Equal to *Sn* if not provided.	0	*MVA*	
wref0	ω_{ref0}	Base speed reference	1	*p.u.*	
R	R	Speed regulation gain (mach. base default)	0.050	*p.u.*	ipower
r	r	Temporary droop (R<r)	1	*p.u.*	ipower
GMAX	G_{max}	Maximum governor response	1	*p.u.*	power
GMIN	G_{min}	Minimum governor response	0	*p.u.*	power
VELM	$VELM$	Gate velocity limit	0.300	*p.u.*	power
Tf	T_f	Filter time constant	0.050		
Tr	T_r	Governor time constant	1		
Tg	T_g	Servo time constant	0.050		
Dt	D_t	Turbine damping coefficient	0		power
qNL	q_{NL}	No-load flow at nominal head	0.100		power
Tw	T_w	Water inertia time constant constant	1		
At	A_t	Turbine gain	1		
dbL	db_L	Lower bound of deadband	0	*p.u.*	
dbU	db_U	Upper bound of deadband	0	*p.u.*	
ue	u_e	effective online status	0		
Sg	S_n	Rated power from generator	0	*MVA*	
Vn	V_n	Rated voltage from generator	0	*kV*	

Variables

Name	Symbol	Type	Description	Unit	Properties
LG_y	LG_y	State	State in lag transfer function		v_str
gtpos	$gtpos$	State	State in gate position (c)	*rad*	v_str
LAG_y	LAG_y	State	State in lag transfer function		v_str
q_y	q_y	State	Integrator output		v_str
omega	ω	ExtState	Generator speed	*p.u.*	
paux	$paux$	Algeb	Auxiliary power input		v_str
pout	$pout$	Algeb	Turbine final output power		v_str
wref	$wref$	Algeb	Speed reference variable		v_str
pref	$pref$	Algeb	Reference power input		v_str
wd	wd	Algeb	Generator speed deviation	*p.u.*	v_str
pd	pd	Algeb	Pref plus speed deviation times gain	*p.u.*	v_str
dg	dg	Algeb	desired gate (c)	*p.u.*	v_str
h	h	Algeb	turbine head	*p.u.*	v_str
DB_y	DB_y	Algeb	Deadband type 1 output		v_str
tm	tm	ExtAlgeb	Mechanical power interface to SynGen		

Initialization Equations

Name	Symbol	Type	Initial Value
LG_y	LG_y	State	pd
gtpos	$gtpos$	State	q_0
LAG_y	LAG_y	State	dg
q_y	q_y	State	q_0
omega	ω	ExtState	
paux	$paux$	Algeb	$paux_0$
pout	$pout$	Algeb	tm_0ue
wref	$wref$	Algeb	$wref_0$
pref	$pref$	Algeb	Rq_0
wd	wd	Algeb	0
pd	pd	Algeb	0
dg	dg	Algeb	q_0
h	h	Algeb	1
DB_y	DB_y	Algeb	$1.0DB_{dbzl}\left(-dbL + wd\right) + 1.0DB_{dbzu}\left(-dbU + wd\right)$
tm	tm	ExtAlgeb	

Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
LG_y	LG_y	State	$-LG_y + pd$	T_f
gtpos	$gtpos$	State	LG_y	
LAG_y	LAG_y	State	$-LAG_y + dg$	T_g
q_y	q_y	State	$1 - \frac{q_y^2}{LAG_y^2}$	T_w
omega	ω	ExtState	0	

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
paux	$paux$	Algeb	$-paux + paux_0$
pout	$pout$	Algeb	$-pout + ue\left(Ath\left(-qNL + q_y\right) - DtLAG_y wd\right)$
wref	$wref$	Algeb	$-wref + wref_0$
pref	$pref$	Algeb	$Rq_0 - pref$
wd	wd	Algeb	$ue\left(\omega - wref\right) - wd$
pd	pd	Algeb	$-pd + ue\left(-DB_y - Rdg + paux + pref\right)$
dg	dg	Algeb	$LG_y gr - dg + gtpos$
h	h	Algeb	$-h + \frac{q_y^2}{LAG_y^2}$
DB_y	DB_y	Algeb	$1.0DB_{dbzl}\left(-dbL + wd\right) + 1.0DB_{dbzu}\left(-dbU + wd\right) - DB_y$
tm	tm	ExtAlgeb	$ue\left(pout - tm_0\right)$

Services

Name	Symbol	Equation	Type
pref0	P_{ref0}	tm_0	ConstService
paux0	P_{aux0}	0	ConstService
VELMn	$-VELM$	$-VELM$	ConstService
tr	$r * Tr$	Trr	ConstService
gr	$1/r$	$\frac{1}{r}$	ConstService
ratel	$rate_l$	$-VELM - gr$	ConstService
rateu	$rate_u$	$VELM - gr$	ConstService
q0	q_0	$qNL + \frac{tm_0}{At}$	ConstService
dgl	dg_{lower}	$-LG_y gr - VELM$	VarService
dgu	dg_{upper}	$-LG_y gr + VELM$	VarService

Discretes

Name	Symbol	Type	Info
dg_lim	lim_{dg}	AntiWindupRate	gate velocity and position limiter
DB_db	db_{DB}	DeadBand	

Blocks

Name	Symbol	Type	Info
LG	LG	Lag	filter after speed deviation (e)
LAG	LAG	Lag	gate opening (g)
q	q	Integrator	turbine flow (q)
DB	DB	DeadBand1	deadband for speed deviation

Setpoints

Name	Target	Info
pref	pref0	initial pref
paux	paux0	const. auxiliary input

Config Fields in [HYGOVDB]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

HYGOV4

HYGOV4 turbine governor model.

Implements the PSS/E HYGOV4 model with the following ignored:

- input deadband DB1
- valve position deadband DB2
- nonlinear function bewteen GV and P_{GV}

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
syn		Synchronous generator idx			mandatory,unique
Tn	T_n	Turbine power rating. Equal to *Sn* if not provided.	0	*MVA*	
wref0	ω_{ref0}	Base speed reference	1	*p.u.*	
Rperm	R_{perm}	Speed Regulation Gain (mach. base default)	0.500	*p.u.*	ipower
Rtemp	R_{temp}	Temporary Droop (Rtemp < Rperm)	1	*p.u.*	ipower
UO	U_O	Maximum Gate opening velocity	1	*p.u.*	power
UC	U_C	Maximum Gate closing velocity	0	*p.u.*	power
PMAX	P_{MAX}	Maximum Gate opening	1	*p.u.*	power
PMIN	P_{MIN}	Minimum Gate opening	0	*p.u.*	power
Tp	T_p	Pilot servo time constant	0.050		
Tg	T_g	Gate servo time constant	0.050		
Tr	T_r	Dashpot time constant	0.050		
Tw	T_w	Water inertia time constant	1		
At	A_t	Turbine gain	1		
Dturb	D_{turb}	Turbine Damping Factor	0		power
Hdam	H_{dam}	Head available at dam	1		power
qNL	q_{NL}	No-Load flow at nominal head	0.100		power
ue	u_e	effective online status	0		
Sg	S_n	Rated power from generator	0	*MVA*	
Vn	V_n	Rated voltage from generator	0	*kV*	

Variables

Name	Symbol	Type	Description	Unit	Properties
GATE_y	$GATE_y$	State	AW Integrator output		v_str
WO_x	WO_x	State	State in washout filter		v_str
LAG_y	LAG_y	State	State in lag transfer function		v_str
q_y	q_y	State	Integrator output		v_str
omega	ω	ExtState	Generator speed	*p.u.*	
paux	$paux$	Algeb	Auxiliary power input		v_str
pout	$pout$	Algeb	Turbine final output power		v_str
wref	$wref$	Algeb	Speed reference variable		v_str
pref	$pref$	Algeb	Reference power input		v_str
wd	wd	Algeb	Generator speed deviation	*p.u.*	v_str
SV_x	SV_x	Algeb	Value before limiter		v_str
WO_y	WO_y	Algeb	Output of washout filter		v_str
Psum	$Psum$	Algeb	summation of power input to servo	*p.u.*	v_str
trhead	$trhead$	Algeb	turbine head	*p.u.*	v_str
tm	tm	ExtAlgeb	Mechanical power interface to SynGen		

Initialization Equations

Name	Symbol	Type	Initial Value
GATE_y	$GATE_y$	State	$\frac{q_0}{Hdam^{0.5}}$
WO_x	WO_x	State	$GATE_y$
LAG_y	LAG_y	State	$Psum$
q_y	q_y	State	q_0
omega	ω	ExtState	
paux	$paux$	Algeb	$paux_0$
pout	$pout$	Algeb	tm_0ue
wref	$wref$	Algeb	$wref_0$
pref	$pref$	Algeb	$\frac{Rpermq_0}{Hdam^{0.5}}$
wd	wd	Algeb	0
SV_x	SV_x	Algeb	LAG_yiTg
WO_y	WO_y	Algeb	0
Psum	$Psum$	Algeb	0
trhead	$trhead$	Algeb	$Hdam$
tm	tm	ExtAlgeb	

Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
GATE_y	$GATE_y$	State	$SV_{limzi}SV_x + SV_{limzl}UC + SV_{limzu}UO$	1
WO_x	WO_x	State	$GATE_y - WO_x$	T_r
LAG_y	LAG_y	State	$-LAG_y + Psum$	T_p
q_y	q_y	State	$Hdam - trhead$	T_w
omega	ω	ExtState	0	

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
paux	$paux$	Algeb	$-paux + paux_0$
pout	$pout$	Algeb	$-pout + ue\left(Attrhead\left(-qNL + q_y\right) - DturbGATE_y wd\right)$
wref	$wref$	Algeb	$-wref + wref_0$
pref	$pref$	Algeb	$\frac{Rpermq_0}{Hdam^{0.5}} - pref$
wd	wd	Algeb	$ue\left(\omega - wref\right) - wd$
SV_x	SV_x	Algeb	$LAG_y iTg - SV_x$
WO_y	WO_y	Algeb	$-TrWO_y + TrRtemp\left(GATE_y - WO_x\right)$
Psum	$Psum$	Algeb	$-Psum + ue\left(-GATE_y Rperm - WO_y + paux + pref - wd\right)$
trhead	$trhead$	Algeb	$-trhead + \frac{q_y^2}{GATE_y^2}$
tm	tm	ExtAlgeb	$ue\left(pout - tm_0\right)$

Services

Name	Symbol	Equation	Type
pref0	P_{ref0}	tm_0	ConstService
paux0	P_{aux0}	0	ConstService
iTg	$1/T_g$	$\frac{u}{Tg}$	ConstService
R	$R_{temp} + R_{perm}$	$Rperm + Rtemp$	ConstService
TrRtemp	$R_{temp} * T_r$	$RtempTr$	ConstService
q0	q_0	$qNL + \frac{tm_0}{AtHdam}$	ConstService

Discretes

Name	Symbol	Type	Info
SV_lim	lim_{SV}	HardLimiter	
GATE_lim	lim_{GATE}	AntiWindup	Limiter in integrator

Blocks

Name	Symbol	Type	Info
SV	SV	GainLimiter	servo gain and limiters
GATE	$GATE$	IntegratorAntiWindup	Gate position
WO	WO	Washout	Washout feedback with T_r
LAG	LAG	Lag	lag block with T_p, outputs velocity
q	q	Integrator	turbine flow (q)

Setpoints

Name	Target	Info
pref	pref0	initial pref
paux	paux0	const. auxiliary input

Config Fields in [HYGOV4]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

5.3.37 Undefined

The undefined group. Holds models with no `group`.

Common Parameters: u, name

5.3.38 VoltComp

Voltage compensator group for synchronous generators.

Common Parameters: u, name, avr, rc, xc

Common Variables: vcomp

Available models: *IEEEVC*

IEEEVC

Voltage compensator IEEEVC model.

Reference:

[1] PowerWorld, Voltage Compensator, IEEEVC, [Online],

[2] NEPLAN, Exciters Models, [Online],

Available:

https://www.powerworld.com/WebHelp/Content/TransientModels_HTML/Voltage%20Compensator%20IEEEVC.htm?TocPath=%7C%7C%7CIEEEVC%7C_____0

https://www.neplan.ch/wp-content/uploads/2015/08/Nep_EXCITERS1.pdf

Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	u	connection status	1	*bool*	
name		device name			
avr		Exciter idx			mandatory
rc	r_c	Active compensation degree.	0		z
xc	x_c	Reactive compensation degree.	0		z
ue	u_e	effective online status	0		
syn		Retrieved generator idx	0		

Variables

Name	Symbol	Type	Description	Unit	Properties
vcomp	$vcomp$	Algeb	Compensator output voltage to exciter		v_str
v	v	ExtAlgeb	Retrieved bus terminal voltage		
vd	vd	ExtAlgeb	d-axis machine voltage		
vq	vq	ExtAlgeb	q-axis machine voltage		
Id	Id	ExtAlgeb	d-axis machine current		
Iq	Iq	ExtAlgeb	q-axis machine current		
Eterm	$Eterm$	ExtAlgeb			v_str

Initialization Equations

Name	Symbol	Type	Initial Value
vcomp	$vcomp$	Algeb	$-uv + vct$
v	v	ExtAlgeb	
vd	vd	ExtAlgeb	
vq	vq	ExtAlgeb	
Id	Id	ExtAlgeb	
Iq	Iq	ExtAlgeb	
Eterm	$Eterm$	ExtAlgeb	$vcomp$

Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
vcomp	$vcomp$	Algeb	$-uev - vcomp + vct$
v	v	ExtAlgeb	0
vd	vd	ExtAlgeb	0
vq	vq	ExtAlgeb	0
Id	Id	ExtAlgeb	0
Iq	Iq	ExtAlgeb	0
Eterm	$Eterm$	ExtAlgeb	$vcomp$

Services

Name	Symbol	Equation	Type
vct	V_{CT}	$u\sqrt{Id^2rc^2 + Id^2xc^2 + 2Idrcvd + 2Idvqxc + Iq^2rc^2 + Iq^2xc^2 + 2Iqrcvq -}$	VarService $\overline{+ vd^2 + vq^2}$

Config Fields in [IEEEVC]

Option	Symbol	Value	Info	Accepted values
allow_adjust		1	allow adjusting upper or lower limits	(0, 1)
adjust_lower		0	adjust lower limit	(0, 1)
adjust_upper		1	adjust upper limit	(0, 1)

5.4 Config Reference

5.4.1 System

Option	Value	Info	Accepted values
freq	60	base frequency [Hz]	float
mva	100	system base MVA	float
diag_eps	0.000	small value for Jacobian diagonals	
warn_abnormal	1	warn initialization out of normal values	(0, 1)

5.4.2 PFlow

Option	Value	Info	Accepted values
linsolve	0	solve symbolic factorization each step (enable when KLU segfaults)	(0, 1)
tol	0.000	convergence tolerance	float
max_iter	25	max. number of iterations	>=10
method	NR	calculation method	('NR', 'dishonest', 'NK')
n_factorize	4	first N iterations to factorize Jacobian in dishonest method	>0
report	1	write output report	(0, 1)
degree	0	use degree in report	(0, 1)
init_tds	0	initialize TDS after PFlow	(0, 1)
linesearch	1	backtracking line search for NR	(0, 1)

5.4.3 TDS

Option	Value	Info
linsolve	0	solve symbolic factorization each step (enable when KLU segfaults)
method	trapezoid	DAE solution method: "trapezoid" and "backeuler" use niter-based step heuristic; "trap_adapt" a
tol	0.000	convergence tolerance
t0	0	simulation starting time
tf	20	simulation ending time
fixt	1	use fixed step size (1) or variable (0); adaptive methods (trap_adapt, qndf) override to 0
shrinkt	1	allow step shrinking on non-convergence when fixt=1
honest	0	honest Newton method that updates Jac at each step
tstep	0.033	integration step size
max_iter	15	maximum number of iterations
refresh_event	0	refresh events at each step
test_init	1	test if initialization passes
check_conn	1	re-check connectivity after event
criteria	1	use criteria to stop simulation if unstable
ddelta_limit	180	delta diff. limit to be considered unstable, in degree
g_scale	1	scale algebraic residuals with time step size
reset_tiny	1	reset tiny residuals to zero to avoid chattering
qrt	0	quasi-real-time stepping
kqrt	1	quasi-real-time scaling factor; kqrt > 1 means slowing down
store_z	0	store limiter status in TDS output
store_f	0	store RHS of diff. equations
store_h	0	store RHS of external diff. equations
store_i	0	store RHS of external algeb. equations
limit_store	0	limit in-memory timeseries storage
max_store	900	maximum steps of data stored in memory before offloading
save_every	1	save one step to memory every N simulation steps
save_mode	auto	automatically or manually save output data when done
no_tqdm	0	disable tqdm progressbar
chatter_iter	4	minimum iterations to detect chattering
linesearch	1	nonmonotone backtracking line search
dtmax	0	maximum step size (0 = auto from freq and tspan)
dtmin_adapt	0	minimum step size for adaptive methods (0 = auto)
abstol	0.000	absolute tolerance for adaptive LTE estimation
reltol	0.001	relative tolerance for adaptive LTE estimation

5.4.4 EIG

Option	Value	Info	Accepted values
linsolve	0	solve symbolic factorization each step (enable when KLU segfaults)	(0, 1)
plot	0	show plot after computation	(0, 1)
tol	0.000	numerical tolerance to treat eigenvalues as zeros	
gy_tol	0.000	row norm threshold for eliminating dead algebraic variables	

5.4.5 SE

Option	Value	Info	Accepted values
linsolve	0	solve symbolic factorization each step (enable when KLU segfaults)	(0, 1)
tol	0.000	convergence tolerance	float
max_iter	100	max number of SE iterations	>=1
flat_start	0	use flat start (1.0 pu, 0 rad) as initial guess	(0, 1)
report	1	write output report	(0, 1)

5.4.6 CPF

Option	Value	Info	Accepted values
linsolve	0	solve symbolic factorization each step (enable when KLU segfaults)	(0, 1)
parameterization	pseudo_arcle	continuation method	('natural', 'arclength', 'pseudo_arclength')
step	0.100	initial continuation step size for lambda	float
step_min	0.000	minimum step size	float
step_max	0.500	maximum step size	float
adapt_step	1	enable adaptive step sizing	(0, 1)
tol	0.000	convergence tolerance for corrector	float
max_iter	20	max corrector (NR) iterations per step	>=1
max_steps	500	max continuation steps	>=1
stop_at	NOSE	termination: 'NOSE', 'FULL', or a float lambda	('NOSE', 'FULL', 'float')
report	1	write output report	(0, 1)

5.5 API reference

5.5.1 System

andes.system	System package.
andes.variables	Variable package with classes for numerical DAE.

andes.system

System package.

Modules

andes.system.codegen	Code generation and pycode loading helpers for System.
andes.system.config_runtime	System config runtime helpers for System.
andes.system.dae_compactor	DAE compaction helpers for System.
andes.system.facade	System class for power system data and methods.
andes.system.helpers	System-level utility functions and internal helpers.
andes.system.registry	Registry loading helpers for System.

andes.system.codegen

Code generation and pycode loading helpers for System.

Functions

import_pycode([user_pycode_path])	Helper function to import generated pycode in the following priority:
reload_submodules(module_name)	Helper function for reloading an existing module and its submodules.

import_pycode

andes.system.codegen.**import_pycode**(*user_pycode_path=None*)

Helper function to import generated pycode in the following priority:

1. a user-provided path from CLI. Currently, this is only for specifying the path to store the generated pycode via `andes prepare`.
2. `~/.andes/pycode`. This is where pycode is stored by default.
3. `<andes_package_root>/pycode`. One can store pycode in the ANDES package folder and ship a full package, which does not require code generation.

reload_submodules

andes.system.codegen.**reload_submodules**(*module_name*)

Helper function for reloading an existing module and its submodules.

It is used to reload the `pycode` module after regenerating code.

Classes

CodegenManager(system)	Manage symbolic code generation, loading and pre-compilation.

andes.system.codegen.CodegenManager

class andes.system.codegen.**CodegenManager**(*system*)

Manage symbolic code generation, loading and pre-compilation.

__init__(*system*)

Methods

precompile([models, nomp, ncpu])	Trigger precompilation for the given models.
prepare([quick, incremental, models, nomp, ncpu])	Generate numerical functions from symbolically defined models.
undill([autogen_stale])	Reload generated function functions, from either the `$HOME/.andes/pycode` folder.

andes.system.config_runtime

System config runtime helpers for System.

Classes

SystemConfigRuntime(system)	Manage system-level configuration bootstrapping and persistence.

andes.system.config_runtime.SystemConfigRuntime

class andes.system.config_runtime.**SystemConfigRuntime**(*system*)

Manage system-level configuration bootstrapping and persistence.

Configuration is resolved in four phases, each with higher priority:

1. `load_rc()` — load from `~/.andes/andes.rc`
2. `merge_file_config()` — merge `_config` section from the case file
3. `apply_cli_overrides()` — apply `config_option` from the CLI
4. `finalize()` — create `system.config`, add defaults, validate

Phases are called in order from *andes.system.facade.System.__init__()*.

__init__(*system*)

Methods

apply_cli_overrides()	Phase 3: Apply config_option from the command line into system._config_object.
collect_config()	Collect config data from models into a ConfigParser.
collect_config_rows()	Collect all config values as a flat list of row dicts for serialization to _config sheets/keys.
configure_numpy([seed, divide, invalid])	Configure NumPy based on Config.
finalize([config])	Phase 4: Create system.config and system.runtime from the resolved _config_object, add defaults, and validate.
load_config_rc([conf_path])	Load config from an rc-formatted file.
load_rc([config_path, default_config])	Phase 1: Resolve the rc file path and load it into system._config_object.
merge_file_config(files)	Phase 2: Extract _config rows from the case file and merge them into system._config_object.
save_config([file_path, overwrite])	Save all system, model, and routine configurations to an rc-formatted file.
set_config([config])	Set configuration for the System object.
update_config_object()	Deprecated since version 2.0.

andes.system.dae_compactor

DAE compaction helpers for System.

Classes

DAECompactor(system)	Handle DAE algebraic-variable compaction for replaced static devices.

andes.system.dae_compactor.DAECompactor

class andes.system.dae_compactor.**DAECompactor**(*system*)

Handle DAE algebraic-variable compaction for replaced static devices.

__init__(*system*)

Methods

compact_dae()	Detect replaced static devices and compact dae.y by removing their algebraic variable slots.

andes.system.facade

System class for power system data and methods.

Classes

ExistingModels()	Storage class for existing models
System([case, name, config, config_path, ...])	System contains models and routines for modeling and simulation.

andes.system.facade.ExistingModels

class andes.system.facade.**ExistingModels**

Storage class for existing models

__init__()

Methods

andes.system.facade.System

class andes.system.facade.**System**(*case: str | None = None, name: str | None = None, config: Dict | None = None, config_path: str | None = None, default_config: bool | None = False, options: Dict | None = None, no_undill: bool | None = False, autogen_stale: bool | None = True, **kwargs*)

System contains models and routines for modeling and simulation.

System contains a several special *OrderedDict* member attributes for housekeeping. These attributes include *models*, *groups*, *routines* and *calls* for loaded models, groups, analysis routines, and generated numerical function calls, respectively.

Parameters

no_undill
[bool, optional, default=False] True to disable the call to `System.undill()` at the end of object creation. False by default.

autogen_stale
[bool, optional, default=True] True to automatically generate code for stale models.

Attributes

dae
[andes.variables.dae.DAE] Numerical DAE storage

files
[andes.variables.fileman.FileMan] File path storage

config
[andes.core.Config] System config storage

models
[OrderedDict] model name and instance pairs

groups
[OrderedDict] group name and instance pairs

routines
[OrderedDict] routine name and instance pairs

Notes

System stores model and routine instances as attributes. Model and routine attribute names are the same as their class names. For example, *Bus* is stored at `system.Bus`, the power flow calculation routine is at `system.PFlow`, and the numerical DAE instance is at `system.dae`. See attributes for the list of attributes.

__init__(*case: str | None = None, name: str | None = None, config: Dict | None = None, config_path: str | None = None, default_config: bool | None = False, options: Dict | None = None, no_undill: bool | None = False, autogen_stale: bool | None = True, **kwargs*)

Methods

add(model_name[, param_dict])	Add a device instance for an existing model.
as_dict([vin, skip_empty])	Return system data as a dict where the keys are model names and values are dicts.
b_update(models)	Call the observable variable update method for models in sequence.
build_ybus()	Build bus admittance matrix by aggregating contributions from all models with `flags.ybus == True`.
calc_pu_coeff()	Perform per unit value conversion.
call_models(method, models, *args, **kwargs)	Call methods on the given models.
check_group_common()	Delegate to *andes.system.registry.RegistryLoader*.
collect_config()	Delegate to *andes.system.config_runtime.SystemConfigRuntime*.
collect_ref()	Collect indices into *BackRef* for all models.
connectivity([info])	Delegate to `ConnMan.check_connectivity()`.
e_clear(models)	Clear equation arrays in DAE and model variables.
f_update(models)	Call the differential equation update method for models in sequence.
fg_to_dae()	Collect equation values into the DAE arrays.
find_connected(model_or_group, idx)	Find all devices connected to a given device.
find_devices()	Add dependent devices for all model based on *DeviceFinder*.
find_models(flag[, skip_zero])	Find models with at least one of the flags as True.
from_ipysheet(model, sheet[, vin])	Set an ipysheet object back to model.
g_islands()	Reset algebraic mismatches for islanded buses.
g_update(models)	Call the algebraic equation update method for models in sequence.
get_status(model_or_group, idx)	Get the effective status of a device.
get_z(models)	Get all discrete status flags in a numpy array.
import_groups()	Delegate to *andes.system.registry.RegistryLoader*.
import_models()	Delegate to *andes.system.registry.RegistryLoader*.
import_routines()	Delegate to *andes.system.registry.RegistryLoader*.
init(models, routine)	Initialize the variables for each of the specified models.
j_islands()	Set gy diagonals to eps for a and v variables of islanded buses.

continues on next page

Table 44 – continued from previous page

`j_update`(models[, info])	Call the Jacobian update method for models in sequence.
`l_update_eq`(models[, init, niter])	Update equation-dependent limiter discrete components by calling `l_check_eq` of models.
`l_update_var`(models[, niter, err])	Update variable-based limiter discrete states by calling `l_update_var` of models.
`link_ext_param`([model])	Retrieve values for `ExtParam` for the given models.
`precompile`([models, nomp, ncpu])	Delegate to *andes.system.codegen.CodegenManager*.
`prepare`([quick, incremental, models, nomp, ncpu])	Delegate to *andes.system.codegen.CodegenManager*.
`propagate_init_status`()	Propagate effective status from parent to child models at init time.
`reload`(case, **kwargs)	Reload a new case in the same System object.
`remove_pycapsule`()	Remove PyCapsule objects in solvers.
`reset`([force])	Reset to the state after reading data and setup (before power flow).
`s_update_post`(models)	Update variable services by calling `s_update_post` of models.
`s_update_var`(models)	Update variable services by calling `s_update_var` of models.
`save_config`([file_path, overwrite])	Delegate to *andes.system.config_runtime.SystemConfigRuntime*.
`set_address`(models)	Set addresses for differential and algebraic variables.
`set_config`([config])	Delegate to *andes.system.config_runtime.SystemConfigRuntime*.
`set_dae_names`(models)	Set variable names for differential and algebraic variables, right-hand side of external equations, and discrete flags.
`set_output_subidx`(models)	Process `andes.models.misc.Output` data and store the sub-indices into `dae.xy`.
`set_status`(model_or_group, idx, value)	Set the online status of a device and propagate to dependents.
`set_var_arrays`(models[, inplace, alloc])	Set arrays (*v* and *e*) for internal variables to access dae arrays in place.
`setup`()	Set up system for studies.
`store_adder_setter`(models)	Store non-inplace adders and setters for variables and equations.
`store_existing`()	Store existing models in *System.existing*.
`store_getters`(models)	Build `_getters` and `_setters` so that `vars_to_models` and `vars_to_dae` work during model initialization.

continues on next page

Table 44 – continued from previous page

`store_no_check_init`(models)	Store differential variables with `check_init == False`.
`store_sparse_pattern`(models)	Collect and store the sparsity pattern of Jacobian matrices.
`store_switch_times`(models[, eps])	Store event switching time in a sorted Numpy array in `System.switch_times` and an OrderedDict `System.switch_dict`.
`summary`()	Delegate to `ConnMan.summary()`.
`supported_models`([export])	Return the support group names and model names in a table.
`switch_action`(models)	Invoke the actions associated with switch times.
`to_ipysheet`(model[, vin])	Return an ipysheet object for editing in Jupyter Notebook.
`undill`([autogen_stale])	Delegate to *andes.system.codegen.CodegenManager*.
`vars_to_dae`(model)	Copy variables values from models to *System.dae*.
`vars_to_models`()	Copy variable values from *System.dae* to models.

andes.system.helpers

System-level utility functions and internal helpers.

Active public utilities:

- `fix_view_arrays`: Restore NumPy view arrays after deserialization.
- `example`: Load a pre-configured IEEE 14-bus example system.
- `import_pycode`: Import generated numerical code.

Internal DAE naming helpers (consumed by *andes.system.facade*):

- `_append_model_name`, `_set_xy_name`, `_set_hi_name`, `_set_z_name`.

Deprecated backward-compat wrappers (will be removed in v3.0):

- `_config_numpy`: use `SystemConfigRuntime.configure_numpy()`.
- `load_config_rc`: use `SystemConfigRuntime.load_config_rc()`.

Functions

example([setup, no_output])	Return an `andes.system.System` object for the `ieee14_linetrip.xlsx` as an example.
fix_view_arrays(system[, models])	Point NumPy arrays without OWNDATA (termed "view arrays" here) to the source array.
import_pycode([user_pycode_path])	Import generated numerical code (pycode).
load_config_rc([conf_path])	Backward-compatible wrapper to `andes.system.config_runtime.SystemConfigRuntime.load_config_rc()`.

example

`andes.system.helpers.`**`example`**(*setup=True*, *no_output=True*, ***kwargs*)

Return an `andes.system.System` object for the `ieee14_linetrip.xlsx` as an example.

This function is useful when a user wants to quickly get a System object for testing.

Returns

System
An example `andes.system.System` object.

fix_view_arrays

`andes.system.helpers.`**`fix_view_arrays`**(*system*, *models=None*)

Point NumPy arrays without OWNDATA (termed "view arrays" here) to the source array.

This function properly sets `v` and `e` arrays of internal variables as views of the corresponding DAE arrays.

Inputs will be refreshed for each model.

Parameters

system
[andes.system.System] System object to be fixed

models
[OrderedDict, optional] Subset of models to fix. Defaults to `system.models` (all models).

import_pycode

andes.system.helpers.**import_pycode**(*user_pycode_path=None*)

Import generated numerical code (pycode).

Wrapper around `andes.system.codegen.import_pycode()`.

load_config_rc

andes.system.helpers.**load_config_rc**(*conf_path=None*)

Backward-compatible wrapper to `andes.system.config_runtime.SystemConfigRuntime.load_config_rc()`.

Deprecated since version 2.0: Will be removed in v3.0.

andes.system.registry

Registry loading helpers for System.

Classes

`RegistryLoader`(system)	Manage imports and registration for groups, models and routines.

andes.system.registry.RegistryLoader

class andes.system.registry.**RegistryLoader**(*system*)

Manage imports and registration for groups, models and routines.

__init__(*system*)

Methods

`check_group_common()`	Check if all group common variables and parameters are met.
`import_groups()`	Import all groups classes defined in `models/group.py`.
`import_models()`	Import and instantiate models as System member attributes.
`import_routines()`	Import routines as defined in `routines/__init__.py`.
`load_all()`	Load groups, models and routines in dependency order.

andes.variables

Variable package with classes for numerical DAE.

Modules

andes.variables.dae
andes.variables.fileman
andes.variables.report

andes.variables.dae

Classes

DAE(system)	Class for storing numerical values of the DAE system, including variables, equations and first order derivatives (Jacobian matrices).
DAETimeSeries([dae])	DAE time series data.

andes.variables.dae.DAE

class andes.variables.dae.**DAE**(*system*)

Class for storing numerical values of the DAE system, including variables, equations and first order derivatives (Jacobian matrices).

Variable values and equation values are stored as `numpy.ndarray`, while Jacobians are stored as `kvxopt.spmatrix`. The defined arrays and descriptions are as follows:

DAE Array	Description
x	Array for state variable values
y	Array for algebraic variable values
z	Array for 0/1 limiter states (if enabled)
f	Array for differential equation derivatives
Tf	Left-hand side time constant array for f
g	Array for algebraic equation mismatches

The defined scalar member attributes to store array sizes are

Scalar	Description
m	The number of algebraic variables/equations
n	The number of state variables/equations
o	The number of limiter state flags

The derivatives of f and g with respect to x and y are stored in four `kvxopt.spmatrix` sparse matrices: **fx**, **fy**, **gx**, and **gy**, where the first letter is the equation name, and the second letter is the variable name.

Notes

DAE in ANDES is defined in the form of

$$
\begin{aligned}
T\dot{x} &= f(x, y) \\
0 &= g(x, y)
\end{aligned}
$$

DAE does not keep track of the association of variable and address. Only a variable instance keeps track of its addresses.

__init__(*system*)

Methods

`alloc_or_extend_names()`	Allocate empty lists for names for the given size.
`build_pattern`(name)	Build sparse matrices with stored patterns.
`clear_arrays()`	Reset equation and variable arrays to empty.
`clear_fg()`	Resets equation arrays to empty.
`clear_ijv()`	Clear stored triplets.
`clear_ts()`	Drop the TimeSeries data and create a new one.
`clear_xy()`	Reset variable arrays to empty.
`clear_z()`	Reset status arrays to empty
`get_name`(arr)	Helper function for geting the list of variable names based on the array name.
`get_size`(name)	Get the size of an array or sparse matrix based on name.
`print_array`(name[, values, tol])	Debug helper to print array values and names.
`request_address`(array_name, ndevice, nvar[, ...])	Interface for requesting addresses for a model.
`reset()`	Reset array sizes to zero and clear all arrays.
`resize_arrays()`	Resize arrays to the new sizes *m* and *n*, and *o*.
`restore_sparse`([names])	Restore all sparse matrices to the sparsity pattern filled with zeros (for variable Jacobian elements) and non-zero constants.
`set_t`(t)	Helper function for setting time in-place.
`store()`	Store values for the current time step to the TimeSeries storage.
`store_sparse_ijv`(name, row, col, val)	Store the sparse pattern triplets.
`write_lst`(lst_path)	Dump the variable name lst file.
`write_npy`(file_path)	Write TDS data into NumPy uncompressed format.
`write_npz`(file_path)	Write TDS data into NumPy compressed format.

Attributes

`fg`	Return a read-only concatenated array of [f, g].
`x_name_output`	Return a list of state var names selected by Output.
`x_tex_name_output`	Return a list of state var LaTeX names selected by Output.
`xy`	Return a read-only concatenated array of [x, y].
`xy_name`	Return a concatenated list of all variable names without format.
`xy_tex_name`	Return a concatenated list of all variable names in LaTeX format.
`xyz`	Return a read-only concatenated array of [x, y, z].
`xyz_name`	Return a concatenated list of all variable names without format.
`xyz_tex_name`	Return a concatenated list of all variable names in LaTeX format.
`y_name_output`	Return a list of algeb var names selected by Output.
`y_tex_name_output`	Return a list of algeb var LaTeX names selected by Output.

andes.variables.dae.DAETimeSeries

class `andes.variables.dae.`**`DAETimeSeries`**(*dae=None*)

DAE time series data.

`__init__`(*dae=None*)

Methods

`get_data`(base_vars, *[, a, rhs])	Get time-series data, either for a variable or for the equation associated with the variable.
`reset`()	Reset the internal storage and erase all data.
`unpack`([df, attr, warn_empty])	Unpack dict-stored data into arrays and/or dataframes.
`unpack_df`(attr)	Construct pandas dataframes.
`unpack_np`(attr[, warn_empty])	Unpack dict data into numpy arrays.

Attributes

df	Short-hand for the xy dataframe.

andes.variables.fileman

Functions

add_suffix(name, suffix)	Add suffix to a file name, preserving any real file extension.

add_suffix

andes.variables.fileman.**add_suffix**(*name*, *suffix*)

Add suffix to a file name, preserving any real file extension.

Parameters

name
[str] File name, with or without extension.

suffix
[str] Suffix to append (e.g., 'out', 'prof', 'eig').

Returns

str
Name with suffix inserted before extension (e.g., 'case_out.xlsx') or appended if no extension (e.g., 'case_out').

Notes

Only treats the trailing part as an extension if it contains at least one letter (e.g., '.xlsx', '.txt'). Numeric suffixes like '.59' in 'case_1.59' are not treated as extensions.

Classes

FileMan([case])	Define a File Manager class for System

andes.variables.fileman.FileMan

class andes.variables.fileman.**FileMan**(*case=None*, ***kwargs*)

Define a File Manager class for System

__init__(*case=None*, ***kwargs*)

Initialize the output file names. For inputs, all absolute paths will be respected. All relative paths are relative to *input_path*.

case: must be full path to case

output: desired name for format conversion output

input_path: default path for input files that only contains file name. If *input_path* is not provided,
it will be derived from the path of *case*.

output_path: path for output files. Default to current working directory where *andes* is invoked.

Methods

get_fullpath([fullname])	Return the original full path if full path is specified, otherwise search in the case file path.
set([case])	Perform the input and output set up.

andes.variables.report

Functions

report_info(system)

report_info

andes.variables.report.**report_info**(*system*)

Classes

Report(system)	Report class to store system static analysis reports

andes.variables.report.Report

class andes.variables.report.**Report**(*system*)

Report class to store system static analysis reports

__init__(*system*)

Methods

update()	Update values based on the requested content
write()	Write report to file.

Attributes

info

5.5.2 Routines

andes.routines	Package for ANDES analysis routines.

andes.routines

Package for ANDES analysis routines.

Modules

andes.routines.adaptive	Shared helpers for adaptive integration methods.
andes.routines.base	Base class for ANDES calculation routines.
andes.routines.cpf	Continuation Power Flow (CPF) routine.
andes.routines.criteria	Stability criteria module.
andes.routines.daeint	Integration methods for DAE.
andes.routines.eig	Module for eigenvalue analysis.
andes.routines.pflow	Module for power flow calculation.
andes.routines.qndf	QNDF: Variable-order quasi-constant-step NDF method (Shampine & Reichelt 1997).
andes.routines.se	Static and dynamic state estimation routine.
andes.routines.tds	ANDES module for time-domain simulation.

andes.routines.adaptive

Shared helpers for adaptive integration methods.

Functions

accept_reject(err_est, h, deltatmax, order)	Shared accept/reject controller for adaptive methods.
candidate_h(err_est, h, order[, safety, ...])	Propose a candidate next step size from normalized error.
check_adaptive_bust(tds)	Shared bust-check for adaptive methods (TrapezoidAdaptive, QNDF).
propose_step_factor(err_est, order[, ...])	Propose a multiplicative step-size factor from a normalized error estimate.
weighted_rms_error(err_vec, x_prev, x_new, ...)	Compute weighted RMS norm of an error vector.

accept_reject

andes.routines.adaptive.**accept_reject**(*err_est*, *h*, *deltatmax*, *order*, *fail_count=0*, *accept_threshold=1.0*, *accept_safety=0.9*, *accept_min_factor=0.2*, *accept_max_factor=2.0*, *reject_safety=0.9*, *reject_min_factor=0.2*, *reject_max_factor=0.9*, *repeat_reject_after=1*, *repeat_reject_factor=0.5*)

Shared accept/reject controller for adaptive methods.

Returns

tuple[bool, float, int]
`(accepted, h_next, fail_count_next)`.

candidate_h

`andes.routines.adaptive.`**`candidate_h`**(*err_est*, *h*, *order*, *safety=0.9*, *min_factor=0.2*, *max_factor=5.0*)

Propose a candidate next step size from normalized error.

check_adaptive_bust

`andes.routines.adaptive.`**`check_adaptive_bust`**(*tds*)

Shared bust-check for adaptive methods (TrapezoidAdaptive, QNDF).

Called after `step()` has written `tds.deltat`. If the step was rejected and deltat has fallen to the adaptive minimum, marks simulation as busted.

propose_step_factor

`andes.routines.adaptive.`**`propose_step_factor`**(*err_est*, *order*, *safety=0.9*, *min_factor=0.2*, *max_factor=5.0*)

Propose a multiplicative step-size factor from a normalized error estimate.

weighted_rms_error

`andes.routines.adaptive.`**`weighted_rms_error`**(*err_vec*, *x_prev*, *x_new*, *abstol*, *reltol*, *err_wt*)

Compute weighted RMS norm of an error vector.

Parameters

err_vec
[np.ndarray] Error estimate vector.

x_prev
[np.ndarray] Previous accepted state.

x_new
[np.ndarray] Candidate new state.

abstol
[float] Absolute tolerance.

reltol
[float] Relative tolerance.

err_wt
[np.ndarray] Pre-allocated work array.

andes.routines.base

Base class for ANDES calculation routines.

Classes

`BaseRoutine`([system, config])	Base routine class.

andes.routines.base.BaseRoutine

class `andes.routines.base.`**`BaseRoutine`**(*system=None*, *config=None*)

Base routine class.

Provides references to system, config, and solver.

`__init__`(*system=None*, *config=None*)

Methods

`doc`([max_width, export])	Routine documentation interface.
`init`()	Routine initialization interface.
`report`(**kwargs)	Report interface.
`run`(**kwargs)	Routine main entry point.
`summary`(**kwargs)	Summary interface

Attributes

`class_name`

andes.routines.cpf

Continuation Power Flow (CPF) routine.

Traces the PV curve from a solved base case toward a target loading using a predictor-corrector method. The predictor uses a tangent vector; the corrector solves a standard power flow at fixed lambda.

Classes

ArcLengthParam()	P = ‖dx‖^2 + dlam^2 - step^2.
CPF([system, config])	Continuation Power Flow routine.
NaturalParam()	P = lam - lam_prev - step.
PseudoArcLengthParam()	P = z' * dx + z_lam * dlam - step.

andes.routines.cpf.ArcLengthParam

class andes.routines.cpf.**ArcLengthParam**

P = ‖dx‖^2 + dlam^2 - step^2. Traces full curve.

__init__(**args*, ***kwargs*)

Methods

constraint(xy, lam, xy_prev, lam_prev, step, ...)	
jacobian(xy, lam, xy_prev, lam_prev, z, nm)	

andes.routines.cpf.CPF

class andes.routines.cpf.**CPF**(*system=None*, *config=None*)

Continuation Power Flow routine.

Traces the nose curve (PV curve) by smoothly increasing load and generation from a solved base case toward a target loading condition.

The continuation parameter lambda interpolates linearly:

```
p0(lam) = p0_base + lam * (p0_target - p0_base)
```

At lambda=0 the system is at the base case; at lambda=1 the system is at the target. The nose point (maximum lambda) gives the steady-state loadability limit.

Examples

Uniform load scaling:

```
ss = andes.load('ieee14.raw')
ss.PFlow.run()
ss.CPF.run(load_scale=2.0)
print(ss.CPF.max_lam)
```

Per-device target:

```
ss.CPF.run(p0_target=my_p_array, q0_target=my_q_array)
```

Reactive power limits are enforced through the PV model's built-in PV-to-PQ conversion. Pass `config_option=["PV.pv2pq=1"]` to `andes.load()` to enable Q-limit checking at each corrector step.

__init__(*system=None*, *config=None*)

Methods

`doc`([max_width, export])	Routine documentation interface.
`init`()	Initialize CPF routine.
`plot`(bus_idx[, fig, ax, show])	Plot PV curve for a specific bus.
`plot_qv`([fig, ax, show])	Plot QV curve from the last `run_qv()` call.
`report`()	Print CPF summary.
`run`([load_scale, p0_target, q0_target, ...])	Run continuation power flow.
`run_qv`(bus_idx[, q_range])	Run QV curve analysis at a specific bus.
`summary`()	Summary interface

Attributes

`class_name`

andes.routines.cpf.NaturalParam

class andes.routines.cpf.**NaturalParam**

P = lam - lam_prev - step. Singular at nose.

__init__(**args*, ***kwargs*)

Methods

`constraint`(xy, lam, xy_prev, lam_prev, step, ...)	
`jacobian`(xy, lam, xy_prev, lam_prev, z, nm)	

andes.routines.cpf.PseudoArcLengthParam

class andes.routines.cpf.**PseudoArcLengthParam**

P = z' * dx + z_lam * dlam - step. Most robust at nose.

__init__(**args*, ***kwargs*)

Methods

`constraint`(xy, lam, xy_prev, lam_prev, step, ...)	
`jacobian`(xy, lam, xy_prev, lam_prev, z, nm)	

andes.routines.criteria

Stability criteria module.

Functions

deltadelta(delta, diff_limit)	Test if a system is stable by comparing the maximum rotor angle difference with a threshold.

deltadelta

andes.routines.criteria.**deltadelta**(*delta*, *diff_limit*)

Test if a system is stable by comparing the maximum rotor angle difference with a threshold.

Returns

bool

True if the system is stable, False otherwise.

andes.routines.daeint

Integration methods for DAE.

Classes

BackEuler()	Backward Euler's integration method.
ImplicitIter()	Base class for implicit iterative methods.
Trapezoid()	Trapezoidal methods.
TrapezoidAdaptive()	Adaptive trapezoid with step-doubling LTE estimation.

andes.routines.daeint.BackEuler

class andes.routines.daeint.**BackEuler**

Backward Euler's integration method.

__init__(**args*, ***kwargs*)

Methods

calc_h(tds)	Default step-size control: niter heuristic with fixt/shrinkt handling.
calc_jac(tds, gxs, gys)	Build full Jacobian matrix Ac for Trapezoid method.
calc_q(x, f, Tf, h, x0, f0)	Calculate the residual of algebraized differential equations.
checkpoint_state(tds)	Snapshot DAE state for rollback.
niter_next_h(niter, h, h_min, h_max)	Niter-based step-size heuristic shared by all methods.
restore_state(tds, state)	Restore DAE state from a checkpoint.
solve_once(tds, h, method)	Run one implicit step with the given method and step size.
step(tds)	Integrate with Implicit Trapezoidal Method (ITM) to the current time.

Attributes

`nolte_event_steps`
`nolte_event_window`
`requires_variable_step`

andes.routines.daeint.ImplicitIter

class `andes.routines.daeint.`**`ImplicitIter`**

Base class for implicit iterative methods.

`__init__`(**args*, ***kwargs*)

Methods

`calc_h`(tds)	Default step-size control: niter heuristic with fixt/shrinkt handling.
`calc_jac`(tds, gxs, gys)	
`calc_q`(x, f, Tf, h, x0, f0)	
`checkpoint_state`(tds)	Snapshot DAE state for rollback.
`niter_next_h`(niter, h, h_min, h_max)	Niter-based step-size heuristic shared by all methods.
`restore_state`(tds, state)	Restore DAE state from a checkpoint.
`solve_once`(tds, h, method)	Run one implicit step with the given method and step size.
`step`(tds)	Integrate with Implicit Trapezoidal Method (ITM) to the current time.

Attributes

`nolte_event_steps`
`nolte_event_window`
`requires_variable_step`

andes.routines.daeint.Trapezoid

class andes.routines.daeint.**Trapezoid**

Trapezoidal methods.

__init__(**args*, ***kwargs*)

Methods

calc_h(tds)	Default step-size control: niter heuristic with fixt/shrinkt handling.
calc_jac(tds, gxs, gys)	Build full Jacobian matrix `Ac` for Trapezoid method.
calc_q(x, f, Tf, h, x0, f0)	Calculate the residual of algebraized differential equations.
checkpoint_state(tds)	Snapshot DAE state for rollback.
niter_next_h(niter, h, h_min, h_max)	Niter-based step-size heuristic shared by all methods.
restore_state(tds, state)	Restore DAE state from a checkpoint.
solve_once(tds, h, method)	Run one implicit step with the given method and step size.
step(tds)	Integrate with Implicit Trapezoidal Method (ITM) to the current time.

Attributes

nolte_event_steps
nolte_event_window
requires_variable_step

andes.routines.daeint.TrapezoidAdaptive

class andes.routines.daeint.**TrapezoidAdaptive**

Adaptive trapezoid with step-doubling LTE estimation.

The LTE estimate is based on the difference between one full step of size `h` and two half-steps of size `h/2`.

__init__(**args*, ***kwargs*)

Methods

`calc_h`(tds)	Step size is set by `step()`.
`calc_jac`(tds, gxs, gys)	Build full Jacobian matrix `Ac` for Trapezoid method.
`calc_q`(x, f, Tf, h, x0, f0)	Calculate the residual of algebraized differential equations.
`checkpoint_state`(tds)	Snapshot DAE state for rollback.
`niter_next_h`(niter, h, h_min, h_max)	Niter-based step-size heuristic shared by all methods.
`restore_state`(tds, state)	Restore DAE state from a checkpoint.
`solve_once`(tds, h, method)	Run one implicit step with the given method and step size.
`step`(tds)	One adaptive trapezoid step.

Attributes

`nolte_event_steps`
`nolte_event_window`
`requires_variable_step`

andes.routines.eig

Module for eigenvalue analysis.

Classes

`EIG`(system, config)	Eigenvalue analysis routine

andes.routines.eig.EIG

class andes.routines.eig.**EIG**(*system*, *config*)

Eigenvalue analysis routine

__init__(*system*, *config*)

Methods

calc_As([dense])	Return state matrix and store to self.As.
calc_eig([As])	Calculate eigenvalues and right eigen vectors.
calc_pfactor([As])	Compute participation factor of states in eigenvalues.
doc([max_width, export])	Routine documentation interface.
export_mat()	Export state matrix to a <CaseName>_As.mat file with the variable name As, where <CaseName> is the test case name.
find_dead_algebs(gy, gx)	Find algebraic variables with near-zero gy rows.
find_zero_states()	Find the indices of states associated with zero time constants in x.
init()	Routine initialization interface.
plot([mu, fig, ax, left, right, ymin, ymax, ...])	Plot utility for eigenvalues in the S domain.
plot_root_loci(results, eig_indices[, ax, ...])	Plot the root loci.
post_process()	Post processing of eigenvalues.
report([x_name])	Save eigenvalue analysis reports.
run(**kwargs)	Run small-signal stability analysis.
stats()	Return statistics of results in a string.
summary()	Print out a summary to logger.info.
sweep(params, idxes, values)	Parameter sweep for root loci plot.

Attributes

class_name

andes.routines.pflow

Module for power flow calculation.

Classes

PFlow([system, config])	Power flow calculation routine.

andes.routines.pflow.PFlow

class andes.routines.pflow.**PFlow**(*system=None*, *config=None*)

Power flow calculation routine.

Power flow analysis currently supports limiting reactive power (needs to to be turned on via *config.pv2pq*) but does not enforce voltage limits.

__init__(*system=None*, *config=None*)

Methods

doc([max_width, export])	Routine documentation interface.
fg_update()	Evaluate the limiters and residual equations.
init()	Initialize variables for power flow.
newton_krylov([verbose])	Full Newton-Krylov method from SciPy.
nr_solve()	Solve the power flow problem using Newton-Raphson iteration.
nr_step()	Solve a single iteration step using the Newton-Raphson method.
report()	Write power flow report to a plain-text file.
run(**kwargs)	Solve the power flow using the selected method.
summary()	Output a summary for the PFlow routine.

Attributes

class_name

andes.routines.qndf

QNDF: Variable-order quasi-constant-step NDF method (Shampine & Reichelt 1997).

Functions

compute_R(k, rho, R)	
compute_U(k, U)	
compute_beta0(k)	
compute_predictor(cache, k)	
error_constant(k)	
rescale_D(cache, rho, k)	
select_order_and_step(cache, h, order, err_est)	Pick optimal (order, step_size) from order-1, order, order+1.
update_D(D, dx_corr, k)	

compute_R

andes.routines.qndf.**compute_R**(*k*, *rho*, *R*)

compute_U

andes.routines.qndf.**compute_U**(*k*, *U*)

compute_beta0

andes.routines.qndf.**compute_beta0**(*k*)

compute_predictor

andes.routines.qndf.**compute_predictor**(*cache*, *k*)

error_constant

andes.routines.qndf.**error_constant**(*k*)

rescale_D

andes.routines.qndf.**rescale_D**(*cache*, *rho*, *k*)

select_order_and_step

andes.routines.qndf.**select_order_and_step**(*cache*, *h*, *order*, *err_est*)

Pick optimal (order, step_size) from order-1, order, order+1.

update_D

andes.routines.qndf.**update_D**(*D*, *dx_corr*, *k*)

Classes

QNDF()	Variable-order (1-5) quasi-constant-step NDF method.
QNDFCache(n, abstol, reltol[, max_order])	Pre-allocated workspace.

andes.routines.qndf.QNDF

class andes.routines.qndf.**QNDF**

Variable-order (1-5) quasi-constant-step NDF method.

__init__(**args*, ***kwargs*)

Methods

calc_h(tds)	Step size is set by step().
step(tds)	One QNDF step.

Attributes

nolte_event_steps
nolte_event_window
requires_variable_step

andes.routines.qndf.QNDFCache

class andes.routines.qndf.**QNDFCache**(*n*, *abstol*, *reltol*, *max_order=5*)

Pre-allocated workspace. Created once at TDS.init().

__init__(*n*, *abstol*, *reltol*, *max_order=5*)

Methods

reset_for_event(x_current)

andes.routines.se

Static and dynamic state estimation routine.

Classes

SE([system, config])	State estimation routine.

andes.routines.se.SE

class andes.routines.se.**SE**(*system=None*, *config=None*)

State estimation routine.

Supports static SE (single snapshot, WLS by default) and will support dynamic SE (time-stepping with EKF/UKF) in a future phase.

Static SE estimates bus voltage magnitudes and angles from a set of measurements. Requires a converged power flow as starting point.

Examples

Basic usage with auto-generated measurements:

```
ss = andes.load('ieee14.raw')
ss.PFlow.run()
ss.SE.run()
```

With custom measurements:

```
from andes.se import Measurements
m = Measurements(ss)
m.add('Bus', 'v', idx=[1, 2, 6, 8], sigma=0.005)
m.add_bus_injection(sigma_p=0.02, sigma_q=0.03)
m.generate_from_pflow(seed=42)
ss.SE.run(measurements=m)
```

__init__(*system=None*, *config=None*)

Methods

`chi_squared_test`([confidence])	Perform chi-squared test on the SE result.
`doc`([max_width, export])	Routine documentation interface.
`init`([measurements, algorithm])	Initialize state estimation.
`report`()	Log a summary of SE results.
`run`([measurements, algorithm])	Run static state estimation.
`summary`([algo])	Print SE configuration summary.

Attributes

`a_est`	Estimated bus voltage angles (radians).
`class_name`	
`v_est`	Estimated bus voltage magnitudes.

andes.routines.tds

ANDES module for time-domain simulation.

Classes

`TDS`([system, config])	Time-domain simulation routine.

andes.routines.tds.TDS

class `andes.routines.tds.`**`TDS`**(*system=None*, *config=None*)

Time-domain simulation routine.

Some cases may be sensitive to large convergence tolerance `config.tol`. If numerical oscillation happens, try reducing `config.tol` to `1e-6`.

`__init__`(*system=None*, *config=None*)

Methods

`calc_h`([resume])	Calculate the time step size during the TDS.
`check_criteria`()	Check stability criteria.
`do_switch`()	Checks if is an event time and perform switch if true.
`doc`([max_width, export])	Routine documentation interface.
`fg_update`(models[, init])	Perform one round of evaluation for one iteration step.
`get_timeseries`(var)	Return time-series data for a variable as a DataFrame.
`init`()	Initialize the status, storage and values for TDS.
`init_resume`()	Initialize a resumed simulation.
`itm_step`()	Integrate one step of size `self.h` using the configured DAE method.
`load_plotter`()	Manually load a plotter into `TDS.plotter`.
`plot`(*args, **kwargs)	Shorthand for `TDS.plt.plot(...)`.
`reinit`()	Reset TDS to post-init state for a new simulation episode.
`report`(**kwargs)	Report interface.
`reset`()	Reset internal states to pre-init condition.
`rewind`(t)	TODO: rewind to a past time.
`run`([no_summary, from_csv])	Run time-domain simulation using numerical integration.
`save_output`([npz])	Save the simulation data into two files: a *.lst* file and a *.npz* file.
`set_method`([name])	Set DAE solution method.
`streaming_init`()	Send out initialization variables and process init from modules.
`streaming_step`()	Sync, handle and streaming for each integration step.
`summary`()	Print out a summary of TDS options to logger.info.
`test_init`()	Test if the TDS initialization is successful.

Attributes

class_name

5.5.3 Plot

andes.plot	The ANDES plotting tool.

andes.plot

The ANDES plotting tool.

Functions

eig_plot(name, args)	
parse_y(y, upper[, lower])	Parse command-line input for Y indices and return a list of indices
set_font([family, size, style, weight])	Sets the font for matplotlib.
set_latex()	Enables LaTeX for matplotlib based on the *with_latex* option and *dvipng* availability.
set_style([style])	Set matplotlib style.
tdsplot(filename, y[, x, to_csv, find, ...])	TDS plot main function based on the new TDSData class.

eig_plot

andes.plot.**eig_plot**(*name*, *args*)

parse_y

andes.plot.**parse_y**(*y*, *upper*, *lower=0*)

Parse command-line input for Y indices and return a list of indices

Parameters

y
[Union[List, Set, Tuple]]

Input for Y indices. Could be single item (with or without colon), or multiple items

upper
[int] Upper limit. In the return list y, y[i] <= uppwer.

lower
[int] Lower limit. In the return list y, y[i] >= lower.

Returns

set_font

`andes.plot.set_font(family='serif', size=12, style='normal', weight='normal')`

Sets the font for matplotlib.

Parameters

family
[str] Font family.

size
[int] Font size.

style
[str] Font style.

weight
[str] Font weight.

set_latex

`andes.plot.set_latex()`

Enables LaTeX for matplotlib based on the *with_latex* option and *dvipng* availability.

Returns

bool
True for LaTeX on, False for off

set_style

`andes.plot.set_style(style='default')`

Set matplotlib style.

Parameters

style
[str] *default*, *ieee* (require *scienceplots*), or other available styles (see *matplotlib.pyplot.style.available*).

tdsplot

andes.plot.**tdsplot**(*filename*, *y*, *x=(0,)*, *to_csv=False*, *find=None*, *xargs=None*, *exclude=None*, ***kwargs*)

TDS plot main function based on the new TDSData class.

Parameters

filename
[str] Path to the ANDES TDS output data file. Works without extension.

x
[list or int, optional] The index for the x-axis variable. x=0 by default for time

y
[list or int] The indices for the y-axis variable

to_csv
[bool] True if need to export to a csv file

find
[str, optional] if not none, specify the variable name to find

xargs
[str, optional] similar to find, but return the result indices with file name, x idx name for xargs

exclude
[str, optional] variable name pattern to exclude

Returns

TDSData object

Classes

TDSData([full_name, mode, dae, path])	A data container for loading and plotting results from Andes time-domain simulation.

andes.plot.TDSData

class andes.plot.**TDSData**(*full_name=None*, *mode='file'*, *dae=None*, *path=None*)

A data container for loading and plotting results from Andes time-domain simulation.

__init__(*full_name=None*, *mode='file'*, *dae=None*, *path=None*)

Methods

`bqplot_data`(xdata, ydata, *[, xheader, ...])	Plot with `bqplot`.
`data_to_df`()	Convert to pandas.DataFrame
`export_csv`([path, idx, header, formatted, ...])	Export to a csv file.
`find`(query[, exclude, formatted, idx_only])	Return variable names and indices matching *query*.
`get_call`([backend])	Get the internal *plot_data* function for the specified backend.
`get_header`(idx[, formatted])	Return a list of the variable names at the given indices.
`get_values`(idx)	Return the variable values at the given indices.
`guess_event_time`()	Guess the event starting time from the input data by checking when the values start to change
`load_dae`()	Load from DAE time series.
`load_lst`()	Load the lst file into internal data structures *_idx*, *_fname*, *_uname*, and counts the number of variables to *nvars*.
`load_npy_or_csv`([delimiter])	Load the npy, zpy or (the legacy) csv file into the internal data structure *self._xy*.
`panoview`(mdl, vars, *[, ncols, idx, a, figsize])	Panoramic view of variables of a given model instance.
`plot`(yidx[, xidx, a, ytimes, ycalc, left, ...])	Entry function for plotting.
`plot_data`(xdata, ydata, *[, xheader, ...])	Plot lines for the supplied data and options.
`plotly_data`(xdata, ydata, *[, xheader, ...])	Plot with Plotly for interactive visualization in Jupyter.
`plotn`(nrows, ncols, yidxes[, xidxes, dpi, ...])	Plot multiple subfigures in one figure.

5.5.4 I/O

andes.io	ANDES input parsers and output formatters.

andes.io

ANDES input parsers and output formatters.

Functions

dump(system, output_format[, full_path, ...])	Dump the System data into the requested output format.
get_output_ext(out_format)	Helper function to get the output extension for the given output format.
guess(system)	Guess the input format based on extension and content.
parse(system)	Parse input file with the given format in *system.files.input_format*.
read_file_like(infile)	Read a file-like object and return a list of splitted lines.

dump

andes.io.**dump**(*system*, *output_format*, *full_path=None*, *overwrite=False*, ***kwargs*)

Dump the System data into the requested output format.

Parameters

system
System object

output_format
[str] Output format name. 'xlsx' will be used if is not an instance of *str*.

Returns

bool
True if successful; False otherwise.

get_output_ext

andes.io.**get_output_ext**(*out_format*)

Helper function to get the output extension for the given output format.

Parameters

out_format
[str] Output format name.

Returns

str
[file extension without dot or empty if not supported]

guess

`andes.io.guess(system)`

Guess the input format based on extension and content.

Also stores the format name to *system.files.input_format*.

Parameters

system
[System] System instance with the file name set to *system.files*

Returns

str
format name

parse

`andes.io.parse(system)`

Parse input file with the given format in *system.files.input_format*.

Returns

bool
True if successful; False otherwise.

read_file_like

`andes.io.read_file_like(infile: str | IOBase)`

Read a file-like object and return a list of splitted lines.

Modules

`andes.io.json`	JSON reader and writer for ANDES.
`andes.io.matpower`	Simple MATPOWER format parser
`andes.io.psse`	PSS/E file parser.
`andes.io.streaming`	Simulation data streaming interface for CURENT DiME2.
`andes.io.txt`	Output TXT file formatter.
`andes.io.xlsx`	Excel reader and writer for ANDES power system parameters

andes.io.json

JSON reader and writer for ANDES.

Functions

read(system, infile)	Read JSON file with ANDES model data into an empty system.
testlines(infile)	
write(system, outfile[, skip_empty, overwrite])	Write loaded ANDES system data into a JSON file.

read

andes.io.json.**read**(*system*, *infile: str | IOBase*)

Read JSON file with ANDES model data into an empty system.

Parameters

system
[System] Empty System instance

infile
[str or io.BaseIO] str: path to the input file; or io.BaseIO: a stream to read from

Returns

System
System instance after succeeded

testlines

andes.io.json.**testlines**(*infile*)

write

andes.io.json.**write**(*system*, *outfile*, *skip_empty=True*, *overwrite=None*, ***kwargs*)

Write loaded ANDES system data into a JSON file.

Parameters

system
[System] A loaded system with parameters

outfile
[str] Path to the output file

skip_empty
[bool] Skip output of empty models (n = 0)

overwrite
[bool] None to prompt for overwrite selection; True to overwrite; False to not overwrite

Returns

bool
True if file written; False otherwise

andes.io.matpower

Simple MATPOWER format parser

Functions

m2mpc(infile)	Parse MATPOWER file and return a dictionary with the data.
mpc2system(mpc, system)	Load an mpc dict into an empty ANDES system.
read(system, file)	Read a MATPOWER data file into mpc, and build andes device elements.
system2mpc(system)	Convert data from an ANDES system to an mpc dict.
testlines(infile)	Test if this file is in the MATPOWER format.

m2mpc

andes.io.matpower.**m2mpc**(*infile: str*) → dict

Parse MATPOWER file and return a dictionary with the data.

Supported fields include

- baseMVA
- bus
- bus_names
- gen
- branch
- gencost (parsed but not used)
- areas (parsed but not used)

Parameters

infile
[str] Path to the MATPOWER file.

Returns

dict
mpc struct names : numpy arrays

mpc2system

`andes.io.matpower.`**`mpc2system`**(*mpc: dict*, *system*) → bool

Load an mpc dict into an empty ANDES system.

Parameters

system
[andes.system.System] Empty system to load the data into.

mpc
[dict] mpc struct names : numpy arrays

Returns

bool
True if successful, False otherwise.

read

`andes.io.matpower.`**`read`**(*system*, *file*)

Read a MATPOWER data file into mpc, and build andes device elements.

system2mpc

`andes.io.matpower.`**`system2mpc`**(*system*) → dict

Convert data from an ANDES system to an mpc dict.

In the `gen` section, slack generators preceeds PV generators.

testlines

`andes.io.matpower.`**`testlines`**(*infile*)

Test if this file is in the MATPOWER format.

NOT YET IMPLEMENTED.

andes.io.psse

PSS/E file parser.

Include a RAW parser and a DYR parser.

Functions

get_block_lines(b, mdata, cfg)	Return the number of lines based on the block index in the RAW file.
read(system, file)	Read PSS/E RAW file v32/v33/v34 formats.
read_add(system, file)	Read an addition PSS/E dyr file.
sort_psse_models(dyr_yaml, system)	Sort supported models so that model names are ordered by dependency.
testlines(infile)	Check the raw file for frequency base.

get_block_lines

andes.io.psse.**get_block_lines**(*b*, *mdata*, *cfg*)

Return the number of lines based on the block index in the RAW file.

read

andes.io.psse.**read**(*system*, *file*)

Read PSS/E RAW file v32/v33/v34 formats.

read_add

andes.io.psse.**read_add**(*system*, *file*)

Read an addition PSS/E dyr file.

Parameters

system
[System] System instance to which data will be loaded

file
[str] Path to the additional *dyr* file

Returns

bool
data parsing status

sort_psse_models

andes.io.psse.**sort_psse_models**(*dyr_yaml*, *system*)

Sort supported models so that model names are ordered by dependency.

Dependency is determined by checking the `find` key in `psse-dyr.yaml` for each model.

Returns

list

The sequence of model names for loading parameters.

testlines

andes.io.psse.**testlines**(*infile*)

Check the raw file for frequency base.

andes.io.streaming

Simulation data streaming interface for CURENT DiME2.

Classes

Streaming(system)	ANDES data streaming class to interface with CURENT LTB.

andes.io.streaming.Streaming

class andes.io.streaming.**Streaming**(*system*)

ANDES data streaming class to interface with CURENT LTB.

__init__(*system*)

Methods

`build_init()`	Build *Varheader*, *Idxvgs* and *SysParam* after power flow routine
`connect()`	Connect to DiME 2 server.
`finalize()`	Send `DONE` signal when simulation completes
`handle_alter`(Alter)	Handle parameter altering
`handle_event`(Event)	Handle Fault, Breaker, Syn and Load Events
`record_module_init`(name, init_var)	Record the variable requests from modules
`send_init`([recepient])	Broadcast *Varheader*, *Idxvgs* and *SysParam* to all DiME clients after power flow routine
`sync_and_handle()`	Sync until the queue is empty.
`transpose_matlab_row`(a)	
`vars_to_modules()`	Stream the results from the last step to modules
`vars_to_pmu()`	Broadcast all PMU measurements and BusFreq measurements in the variable *pmudata*

andes.io.txt

Output TXT file formatter.

Functions

`dump_data`(text, header, rowname, data, file)

dump_data

`andes.io.txt.`**`dump_data`**(*text*, *header*, *rowname*, *data*, *file*, *width=18*, *precision=5*)

andes.io.xlsx

Excel reader and writer for ANDES power system parameters

This module utilizes openpyxl, xlsxwriter and pandas.Frame.

While I like the simplicity of the dome format, spreadsheets are easier to view and edit.

Functions

read(system, infile)	Read an xlsx file with ANDES model data into an empty system
testlines(infile)	
write(system, outfile[, skip_empty, ...])	Write loaded ANDES system data into an xlsx file

read

andes.io.xlsx.**read**(*system*, *infile*)

Read an xlsx file with ANDES model data into an empty system

Parameters

system
[System] Empty System instance

infile
[str or file-like] Path to the input file, or a file-like object

Returns

System
System instance after succeeded

testlines

andes.io.xlsx.**testlines**(*infile*)

write

andes.io.xlsx.**write**(*system*, *outfile*, *skip_empty=True*, *overwrite=None*, *add_book=None*, ***kwargs*)

Write loaded ANDES system data into an xlsx file

Parameters

system
[System] A loaded system with parameters

outfile
[str] Path to the output file

skip_empty
[bool] Skip output of empty models (n = 0)

overwrite
[bool, optional] None to prompt for overwrite selection; True to overwrite; False to not overwrite

add_book
[str, optional] An optional model to be added to the output spreadsheet

Returns

bool
True if file written; False otherwise

5.5.5 Interoperability

andes.interop	Interopability package between Andes and other software.

andes.interop

Interopability package between Andes and other software.

To install dependencies, do:

```
pip install andes[interop]
```

To install dependencies for *development*, in the ANDES source code folder, do:

```
pip install -e .[interop]
```

Modules

andes.interop.gridcal	Basic GridCal (4.6.1) interface, based on the pandapower interface written by Jinning Wang
andes.interop.matpower	Interoperability with MATPOWER through `matpower-pip`.
andes.interop.pandapower	Simple pandapower interface.
andes.interop.pypowsybl	Interoperability with pypowsybl.

andes.interop.gridcal

Basic GridCal (4.6.1) interface, based on the pandapower interface written by Jinning Wang

Author: Josep Fanals (@JosepFanals)

Functions

require_gridcal(f)	Decorator for functions that require GridCal.
to_gridcal(ssa[, verify, tol])	Convert an ANDES system to a GridCal grid.

require_gridcal

andes.interop.gridcal.**require_gridcal**(*f*)

Decorator for functions that require GridCal.

to_gridcal

andes.interop.gridcal.**to_gridcal**(*ssa*, *verify=True*, *tol=1e-06*)

Convert an ANDES system to a GridCal grid.

Parameters

ssa
[andes.system.System] The ANDES system to be converted

verify
[bool] If True, the converted network will be verified with the source ANDES system using AC power flow.

tol
[float] The tolerance of error when comparing power flow solutions.

Returns

GridCal.Engine.Core.multi_circuit.MultiCircuit
A GridCal net with the same bus, branch, gen, and load data as the ANDES system

Notes

Handling of the following parameters:

- By default, all generators in `ssp` are controllable unless user-defined controllability is given
- The online status of generators are determined by the online status of `StaticGen` that connected to the `SynGen` or `DG`
- `ssp.gen.name` is from `ssa.StaticGen.idx`, which should be unique

andes.interop.matpower

Interoperability with MATPOWER through `matpower-pip`.

Please install the Python package `matpower` and configure MATLAB or Octave, following the instructions at matpower-pip.

To create a MATLAB/Octave instance, do:

```
from andes.interop.matpower import start_instance

m = start_instance()
```

Functions

from_matpower(m, varname[, system])	Retrieve a MATPOWER mpc case from a MATLAB/Octave instance.
require_matpower(f)	Decorator for functions that require matpower.
to_matpower(m, varname, system)	Send an ANDES case to a running MATLAB instance.

from_matpower

`andes.interop.matpower.`**`from_matpower`**(*m*, *varname*, *system=None*)

Retrieve a MATPOWER mpc case from a MATLAB/Octave instance.

Parameters

m
[MATLAB/Octave instance] Instance from which to retrieve the MATPOWER case.

varname
[str] Name of the variable in the MATPOWER MPC format to retrieve.

Returns

System
System from the mpc case. The system will not have been set up.

Examples

To retrieve a case from MATPOWER from instance `m`, do the following:

```
from andes.interop.matpower import start_instance, to_matpower, from_
↪matpower

system = from_matpower(m, 'mpc')
```

One can create an Excel file with dynamic data only and use the `xlsx` parser to load data into `system`:

```
from andes.io import xlsx

xlsx.read(system, andes.get_case('ieee14/ieee14_dyn_only.xlsx'))
```

The `ieee14_dyn_only.xlsx` is an example spreadsheet that only contains dynamic components. One will need to create the `idx` correctly for dynamic components to match these from the MATPOWER case. If not sure about the indices, one can save the ANDES system to an Excel file, using:

```
xlsx.write(system, 'system_static.xlsx')
```

require_matpower

`andes.interop.matpower.`**`require_matpower`**(*f*)

Decorator for functions that require matpower.

to_matpower

`andes.interop.matpower.`**`to_matpower`**(*m*, *varname*, *system*)

Send an ANDES case to a running MATLAB instance.

Parameters

m
[MATLAB/Octave instance] Instance to which to send the MATPOWER case.

varname
[str] Name of the variable to store the mpc case in MATLAB/Octave.

system
[`System`] System whose power flow data to send to MATPOWER.

Examples

The code below will create an IEEE 14-bus example system in ANDES, convert it to MATPOWER's case, and send to the MATLAB/Octave instance.

```
import andes

from andes.interop.matpower import (start_instance,
    to_matpower, from_matpower)

m = start_instance()

ss = andes.system.example()
mpc = to_matpower(m, 'mpc', ss)

m.eval("runpf(mpc)")

mpc_out = m.pull("mpc")  # retrieve the mpc case from MATLAB/Octave
```

andes.interop.pandapower

Simple pandapower interface.

Functions

add_gencost(ssp, gen_cost)	Add cost function to converted pandapower net *ssp*.
build_group_table(ssa, group, columns[, ...])	Build the table for devices in a group in an ANDES System.
make_GSF(ppn[, verify, using_sparse_solver])	Build the Generation Shift Factor matrix of a pandapower net.
make_link_table(ssa)	Build the link table for generators and generator controllers in an ANDES System, including `SynGen` and `DG` for now.
require_pandapower(f)	Decorator for functions that require pandapower.
runopp_map(ssp, link_table, **kwargs)	Run OPF in pandapower using `pp.runopp()` and map results back to ANDES based on the link table.
to_pandapower(ssa[, ctrl, verify, tol])	Convert an ANDES system to a pandapower network for power flow studies.

add_gencost

`andes.interop.pandapower.add_gencost(ssp, gen_cost)`

Add cost function to converted pandapower net *ssp*.

The cost data follows the same format of pypower and matpower.

Now only poly_cost is supported.

Parameters

ssp
The pandapower net

gen_cost
[array] generator cost data

build_group_table

`andes.interop.pandapower.build_group_table(ssa, group, columns, mdl_name=None)`

Build the table for devices in a group in an ANDES System.

Parameters

ssa
[andes.system.System] The ANDES system to build the table

group
[string] The ANDES group

columns
[list of string] The common columns of a group that to be included in the table.

mdl_name
[list of string] The list of models that to be included in the table. Default as all models.

Returns

DataFrame
The output Dataframe contains the columns from the device

make_GSF

`andes.interop.pandapower.make_GSF(ppn, verify=True, using_sparse_solver=False)`

Build the Generation Shift Factor matrix of a pandapower net.

Parameters

ppn
[pandapower.network.Network] Pandapower network

verify
[bool] True to verify the GSF with that from DC power flow

using_sparse_solver
[bool] True to use a sparse solver for pandapower maktPTDF

Returns

np.ndarray
The GSF array

make_link_table

`andes.interop.pandapower.`**`make_link_table`**(*ssa*)

Build the link table for generators and generator controllers in an ANDES System, including `SynGen` and `DG` for now.

Parameters

ssa
[andes.system.System] The ANDES system to link

Returns

DataFrame
Each column in the output Dataframe contains the `idx` of linked `StaticGen`, `Bus`, `DG`, `RenGen`, `RenExciter`, `SynGen`, `Exciter`, and `TurbineGov`, `gammap`, `gammaq`.

require_pandapower

`andes.interop.pandapower.`**`require_pandapower`**(*f*)

Decorator for functions that require pandapower.

runopp_map

`andes.interop.pandapower.`**`runopp_map`**(*ssp*, *link_table*, ***kwargs*)

Run OPF in pandapower using `pp.runopp()` and map results back to ANDES based on the link table.

Parameters

ssp
[pandapower network] The pandapower network

link_table
[DataFrame] The link table of ANDES system

Returns

DataFrame
The DataFrame contains the OPF results with columns `p_mw`, `q_mvar`, `vm_pu` in p.u., and the corresponding `idx` of `StaticGen`, `Exciter`, `TurbineGov` in ANDES.

Notes

- The pandapower net and the ANDES system must have same base MVA.
- Multiple `DG` connected to the same `StaticGen` will be converted to one generator. The power is dispatched to each `DG` by the power ratio `gammap`

to_pandapower

`andes.interop.pandapower.`**`to_pandapower`**(*ssa*, *ctrl=None*, *verify=True*, *tol=1e-06*)

Convert an ANDES system to a pandapower network for power flow studies.

Parameters

ssa
[andes.system.System] The ANDES system to be converted

ctrl
[list] The controlability of generators. The length should be the same with the number of `StaticGen`. If not given, controllability of generators will be assigned by default. Example input: [1, 0, 1, ...]; `PV` first, then `Slack`.

verify
[bool] If True, the converted network will be verified with the source ANDES system using AC power flow.

tol
[float] The tolerance of error.

Returns

pandapower.auxiliary.pandapowerNet
A pandapower net with the same bus, branch, gen, and load data as the ANDES system

Notes

Handling of the following parameters:

- This interface tracks static power flow model in ANDES: *Bus*, *Line*, *PQ*, *Shunt*, *PV*, and *Slack*. However, it does not track the dynamic models in ANDES, including but not limited to *TurbineGov*, *SynGen*, and *Exciter*.
- The interface converts the `Slack` in ANDES to `gen` in pandapower rather than `ext_gen`.

- MUST NOT verify power flow after initializing TDS in ANDES. ANDES does not allow running `PFlow` for systems with initialized TDS as it will break variable addressing.
- If you want to track dynamic model outputs in ANDES and feedback into pandapower, you might need to manually transfer the results from ANDES to pandapower.
- Generator cost is not included in the conversion. Use `add_gencost()` to add cost data.
- By default, all generators in `ssp` are controllable unless user-defined controllability is given
- The online status of generators are determined by the online status of `StaticGen` that connected to the `SynGen` or `DG`
- `ssp.gen.name` is from `ssa.StaticGen.idx`, which should be unique

andes.interop.pypowsybl

Interoperability with pypowsybl.

Functions

require_pypowsybl(f)	Decorator for functions that require pypowsybl.
to_pypowsybl(ss)	Convert an ANDES system to a pypowsybl network.

require_pypowsybl

`andes.interop.pypowsybl.`**`require_pypowsybl`**(*f*)

Decorator for functions that require pypowsybl.

to_pypowsybl

`andes.interop.pypowsybl.`**`to_pypowsybl`**(*ss*)

Convert an ANDES system to a pypowsybl network.

Parameters

ss

[andes.system.System] The ANDES system to be converted.

Returns

pypowsybl.network.Network

> **Warning:**
>
> - Power flow results are not verified.

Notes

- Only the BUS_BREAKER topology is supported.
- Each bus has a voltage level named "VL" followed by the bus idx.
- Buses connected by transformers are in the same substation.

Examples

One can utilize pypowsybl to draw network topology. For example,

```
ss = andes.system.example()
n = to_pypowsybl(ss)
results = pp.loadflow.run_ac(n)

n.get_network_area_diagram()  # show diagram for system
n.get_single_line_diagram("VL6")  # show single-line diagram for bus 6
```

5.5.6 Others

andes.cli	ANDES command-line interface and argument parsers.
andes.main	Main entry point for the ANDES CLI and scripting interfaces.
andes.utils.paths	Utility functions for loading andes stock test cases
andes.utils.snapshot	Utility functions for saving and loading snapshots.
andes.utils.widgets	Support for Jupyter widgets.

andes.cli

ANDES command-line interface and argument parsers.

Functions

create_parser()	Create a parser for the command-line interface.
main()	Entry point of the ANDES command-line interface.
preamble()	Log the ANDES command-line preamble at the *logging.INFO* level

create_parser

`andes.cli.create_parser()`

Create a parser for the command-line interface.

Returns

argparse.ArgumentParser
Parser with all ANDES options

main

`andes.cli.main()`

Entry point of the ANDES command-line interface.

preamble

`andes.cli.preamble()`

Log the ANDES command-line preamble at the *logging.INFO* level

andes.main

Main entry point for the ANDES CLI and scripting interfaces.

Functions

`config_logger`([stream_level, stream, file, ...])	Configure an ANDES logger with a *FileHandler* and a *StreamHandler*.
`demo`(**kwargs)	TODO: show some demonstrations from CLI.
`doc`([attribute, list_supported, config])	Quick documentation from command-line.
`edit_conf`([edit_config])	Edit the Andes config file which occurs first in the search path.
`find_log_path`(lg)	Find the file paths of the FileHandlers.
`load`(case[, codegen, setup, use_input_path])	Load a case and set up a system without running routine.
`misc`([edit_config, save_config, ...])	Miscellaneous commands.
`plot`(*args, **kwargs)	Wrapper for the plot tool.
`prepare`([quick, incremental, models, ...])	Run code generation.
`print_license`()	Print out Andes license to stdout.
`remove_output`([recursive])	Remove the outputs generated by Andes, including power flow reports `_out.txt`, time-domain list `_out.lst` and data `_out.dat`, eigenvalue analysis report `_eig.txt`.
`run`(filename[, input_path, verbose, ...])	Entry point to run ANDES routines.
`run_case`(case, *[, routine, profile, ...])	Run single simulation case for the given full path.
`save_conf`([config_path, overwrite])	Save the Andes config to a file at the path specified by `save_config`.
`selftest`([quick, extra])	Run unit tests.
`set_logger_level`(lg, type_to_set, level)	Set logging level for the given type of handler.
`versioninfo`()	Print version info for ANDES and dependencies.

config_logger

`andes.main.`**`config_logger`**(*stream_level=20*, *, *stream=True*, *file=True*, *log_file='andes.log'*, *log_path=None*, *file_level=10*)

Configure an ANDES logger with a *FileHandler* and a *StreamHandler*.

This function is called at the beginning of `andes.main.main()`. Updating `stream_level` and `file_level` is now supported.

Parameters

stream
[bool, optional] Create a *StreamHandler* for *stdout* if `True`. If `False`, the handler will not be created.

file
[bool, optionsl] True if logging to `log_file`.

log_file
[str, optional] Logg file name for *FileHandler*, `'andes.log'` by default. If `None`,

the *FileHandler* will not be created.

log_path
[str, optional] Path to store the log file. By default, the path is generated by get_log_dir() in utils.misc.

stream_level
[{10, 20, 30, 40, 50}, optional] *StreamHandler* verbosity level.

file_level
[{10, 20, 30, 40, 50}, optional] *FileHandler* verbosity level.

Returns

None

demo

`andes.main.`**`demo`**(***kwargs*)

TODO: show some demonstrations from CLI.

doc

`andes.main.`**`doc`**(*attribute=None*, *list_supported=False*, *config=False*, ***kwargs*)

Quick documentation from command-line.

edit_conf

`andes.main.`**`edit_conf`**(*edit_config: str | bool | None = ''*)

Edit the Andes config file which occurs first in the search path.

Parameters

edit_config
[bool] If `True`, try to open up an editor and edit the config file. Otherwise returns.

Returns

bool
`True` is a config file is found and an editor is opened. `False` if `edit_config` is False.

find_log_path

`andes.main.find_log_path(lg)`

Find the file paths of the FileHandlers.

load

`andes.main.load(case, codegen=False, setup=True, use_input_path=True, **kwargs)`

Load a case and set up a system without running routine. Return a system.

Takes other kwargs recognizable by `System`, such as `addfile`, `input_path`, and `no_putput`.

Parameters

case: str
Path to the test case

codegen
[bool, optional] Call full *System.prepare* on the returned system. Set to True if one need to inspect pretty-print equations and run simulations.

setup
[bool, optional] Call *System.setup* after loading

use_input_path
[bool, optional] True to use the `input_path` argument to behave the same as `andes.main.run`.

> **Warning:** If one need to add devices in addition to these from the case file, do `setup=False` and call `System.add()` to add devices. When done, manually invoke `setup()` to set up the system.

misc

`andes.main.misc(edit_config='', save_config='', show_license=False, clean=True, recursive=False, overwrite=None, version=False, **kwargs)`

Miscellaneous commands.

plot

`andes.main.plot`(**args*, ***kwargs*)

Wrapper for the plot tool.

prepare

`andes.main.prepare`(*quick=False*, *incremental=False*, *models=None*, *precompile=False*, *nomp=False*, ***kwargs*)

Run code generation.

Parameters

full
[bool] True to run full prep with formatted equations. Useful in interactive mode and during document generation.

ncpu
[int] Number of cores to be used for parallel processing.

cli
[bool] True to indicate running from CLI. It will set *quick* to True if not *full*.

precompile
[bool] True to compile model function calls after code generation.

Returns

System object if *cli* is *False*; exit_code 0 otherwise.

> **Warning:** The default behavior has changed since v1.0.8: when *cli* is *True* and *full* is not *True*, quick code generation will be used.

print_license

`andes.main.print_license`()

Print out Andes license to stdout.

remove_output

`andes.main.remove_output(recursive=False)`

Remove the outputs generated by Andes, including power flow reports `_out.txt`, time-domain list `_out.lst` and data `_out.dat`, eigenvalue analysis report `_eig.txt`.

Parameters

recursive
[bool] Recursively clean all subfolders

Returns

bool
`True` is the function body executes with success. `False` otherwise.

run

`andes.main.run(filename, input_path='', verbose=20, mp_verbose=30, ncpu=2, pool=False, cli=False, codegen=False, shell=False, **kwargs)`

Entry point to run ANDES routines.

Parameters

filename
[str] file name (or pattern)

input_path
[str, optional] input search path

verbose
[int, 10 (DEBUG), 20 (INFO), 30 (WARNING), 40 (ERROR), 50 (CRITICAL)] Verbosity level. If `config_logger` is called prior to `run`, this option will be ignored.

mp_verbose
[int] Verbosity level for multiprocessing tasks

ncpu
[int, optional] Number of cpu cores to use in parallel

pool: bool, optional
Use Pool for multiprocessing to return a list of created Systems.

kwargs
Other supported keyword arguments

cli
[bool, optional] If is running from command-line. If True, returns exit code instead of System

codegen
[bool, optional] Run full code generation for System before loading case. Only used for single test case.

shell
[bool, optional] If True, enter IPython shell after routine.

Returns

System or exit_code
An instance of system (if *cli* == *False*) or an exit code otherwise..

run_case

`andes.main.run_case`(*case*, *, *routine='pflow'*, *profile=False*, *convert=''*, *convert_all=''*, *add_book=None*, *codegen=False*, *autogen_stale=True*, *remove_pycapsule=False*, **kwargs*)

Run single simulation case for the given full path. Use `run` instead of `run_case` whenever possible.

Argument `input_path` will not be prepended to `case`.

Arguments recognizable by `load` can be passed to `run_case`.

Parameters

case
[str] Full path to the test case

routine
[str, ('pflow', 'tds', 'eig')] Computation routine to run

profile
[bool, optional] True to enable profiler

convert
[str, optional] Format name for case file conversion.

convert_all
[str, optional] Format name for case file conversion, output sheets for all available devices.

add_book
[str, optional] Name of the device to be added to an excel case as a new sheet.

codegen
[bool, optional] True to run codegen

autogen_stale
[bool, optional] True to automatically generate code for stale models

remove_pycapsule
[bool, optional] True to remove pycapsule from C libraries. Useful when dill serialization is needed.

save_conf

andes.main.**save_conf**(*config_path=None*, *overwrite=None*, ***kwargs*)

Save the Andes config to a file at the path specified by `save_config`. The save action will not run if `save_config = ''`.

Parameters

config_path
[None or str, optional, ('' by default)] Path to the file to save the config file. If the path is an emtpy string, the save action will not run. Save to *~/.andes/andes.conf* if `None`.

Returns

bool
`True` is the save action is run. `False` otherwise.

selftest

andes.main.**selftest**(*quick=False*, *extra=False*, ***kwargs*)

Run unit tests.

set_logger_level

andes.main.**set_logger_level**(*lg*, *type_to_set*, *level*)

Set logging level for the given type of handler.

versioninfo

andes.main.**versioninfo**()

Print version info for ANDES and dependencies.

Classes

TqdmStreamHandler([stream])	StreamHandler that uses tqdm.write() to avoid corrupting progress bars.

andes.main.TqdmStreamHandler

class andes.main.**TqdmStreamHandler**(*stream=None*)

StreamHandler that uses tqdm.write() to avoid corrupting progress bars.

__init__(*stream=None*)

Initialize the handler.

If stream is not specified, sys.stderr is used.

Methods

acquire()	Acquire the I/O thread lock.
addFilter(filter)	Add the specified filter to this handler.
close()	Tidy up any resources used by the handler.
createLock()	Acquire a thread lock for serializing access to the underlying I/O.
emit(record)	Emit a record.
filter(record)	Determine if a record is loggable by consulting all the filters.
flush()	Flushes the stream.
format(record)	Format the specified record.
get_name()	
handle(record)	Conditionally emit the specified logging record.
handleError(record)	Handle errors which occur during an emit() call.
release()	Release the I/O thread lock.
removeFilter(filter)	Remove the specified filter from this handler.
setFormatter(fmt)	Set the formatter for this handler.
setLevel(level)	Set the logging level of this handler.
setStream(stream)	Sets the StreamHandler's stream to the specified value, if it is different.
set_name(name)	

Attributes

name
terminator

andes.utils.paths

Utility functions for loading andes stock test cases

Functions

andes_root()	Return the path to the folder of the ANDES package.
cases_root()	Return the root path to the stock cases
confirm_overwrite(outfile[, overwrite])	Confirm overwriting a file.
get_case(rpath[, check])	Return the path to a stock case for a given path relative to `andes/cases`.
get_config_path([file_name])	Return the path of the config file to be loaded.
get_dot_andes_path()	Return the path to `$HOME/.andes`
get_log_dir()	Get the directory for log file.
get_pycode_path([pycode_path, mkdir])	Get the path to the `pycode` folder.
list_cases([rpath, no_print])	List stock cases under a given folder relative to `andes/cases`
tests_root()	Return the root path to the stock cases

andes_root

`andes.utils.paths.`**`andes_root`**`()`

Return the path to the folder of the ANDES package.

cases_root

`andes.utils.paths.`**`cases_root`**`()`

Return the root path to the stock cases

confirm_overwrite

`andes.utils.paths.`**`confirm_overwrite`**`(`*outfile*, *overwrite=None*`)`

Confirm overwriting a file.

get_case

`andes.utils.paths.get_case(rpath, check=True)`

Return the path to a stock case for a given path relative to `andes/cases`.

To list all cases, use `andes.list_cases()`.

Parameters

check
[bool] True to check if file exists

Examples

To get the path to the case *kundur_full.xlsx* under folder *kundur*, do

```
andes.get_case('kundur/kundur_full.xlsx')
```

get_config_path

`andes.utils.paths.get_config_path(file_name='andes.rc')`

Return the path of the config file to be loaded.

Search Priority: 1. current directory; 2. home directory.

Parameters

file_name
[str, optional] Config file name with the default as `andes.rc`.

Returns

Config path in string if found; None otherwise.

get_dot_andes_path

`andes.utils.paths.get_dot_andes_path()`

Return the path to `$HOME/.andes`

get_log_dir

andes.utils.paths.**get_log_dir**()

Get the directory for log file.

The default is `<tempdir>/andes`, where `<tempdir>` is provided by `tempfile.gettempdir()`.

Returns

str
The path to the temporary logging directory

get_pycode_path

andes.utils.paths.**get_pycode_path**(*pycode_path=None*, *mkdir=False*)

Get the path to the `pycode` folder.

list_cases

andes.utils.paths.**list_cases**(*rpath='.'*, *no_print=False*)

List stock cases under a given folder relative to `andes/cases`

tests_root

andes.utils.paths.**tests_root**()

Return the root path to the stock cases

Classes

DisplayablePath(path, parent_path, is_last)

andes.utils.paths.DisplayablePath

class andes.utils.paths.**DisplayablePath**(*path*, *parent_path*, *is_last*)

__init__(*path*, *parent_path*, *is_last*)

Methods

`displayable()`	
`make_tree`(root[, parent, is_last, criteria])	

Attributes

`display_filename_prefix_last`
`display_filename_prefix_middle`
`display_parent_prefix_last`
`display_parent_prefix_middle`
`displayname`

andes.utils.snapshot

Utility functions for saving and loading snapshots.

Code Examples:

1. Setup base case and save the snapshot for once:

```
import andes

ss = andes.run(andes.get_case("ieee14/ieee14_linetrip.xlsx"))
ss.Toggle.u.v[:] = 0  # turn off line trips for the base case
xy = ss.TDS.init()

andes.utils.snapshot.save_ss("ieee14_snapshot.pkl", ss)
```

2. For every scenario afterwards, load the snapshot and apply disturbances:

```
import andes

ss = andes.utils.snapshot.load_ss("ieee14_snapshot.pkl")

# apply specific disturbances
ss.GENROU.omega.v[0] = 1.02

ss.TDS.run()
```

Functions

load_ss(path)	Load an ANDES snapshot and return a System object.
save_ss(path, system)	Save a system with all internal states as a snapshot.

load_ss

andes.utils.snapshot.**load_ss**(*path*)

Load an ANDES snapshot and return a System object.

Parameters

path
[str] Path to the snapshot file.

Returns

andes.system.System
The loaded system object

save_ss

andes.utils.snapshot.**save_ss**(*path*, *system*)

Save a system with all internal states as a snapshot.

Returns

Path to the saved snapshot.

> **Warning:** One limitation of the current implementation is version dependency. The snapshots only work with the specific ANDES version that created it.

andes.utils.widgets

Support for Jupyter widgets.

Please manually install the following dependencies:

- ipywidgets
- ipysheet

If you are using JupyterLab, do

```
jupyter labextension install @jupyter-widgets/jupyterlab-manager
```

Functions

edit_sheet(system, model)	Use ipysheet to edit parameters of one model.
edit_system(system)	Edit a loaded ANDES System with ipywidgets.
on_close(b)	Callback for the Close botton.
on_update(b)	Callback for the Update button.

edit_sheet

andes.utils.widgets.**edit_sheet**(*system*, *model: str*)

Use ipysheet to edit parameters of one model.

edit_system

andes.utils.widgets.**edit_system**(*system*)

Edit a loaded ANDES System with ipywidgets.

on_close

andes.utils.widgets.**on_close**(*b*)

Callback for the Close botton. Closes ipywidget objects.

on_update

andes.utils.widgets.**on_update**(*b*)

Callback for the Update button. Sets new parameters back to System.

5.6 Release notes

The APIs before v3.0.0 are in beta and may change without prior notice.

5.6.1 v2.0 Notes

v2.0.0 (2026-03-11)

Modeling framework changes:

- Add universal `ue` (effective online status) field to the base `Model` class. Models should use `ue` instead of `u` in equations so that parent-offline status automatically disables children.

- Add `status_parent=True` flag to `IdxParam` for declaring parent-child status relationships (e.g., Exciter->SynGen, PSS->Exciter).
- Add `System.set_status()` / `get_status()` API for setting device status with recursive propagation of `ue = u * parent.ue` to downstream controllers via BackRef.
- Add `propagate_init_status()` for init-time status propagation, replacing the old per-model `ug`/`uee` ExtParam mechanism.
- Add DAE compaction to remove replaced static generators' algebraic variables from the Newton-Raphson system during TDS, reducing system size for dynamics-heavy cases.
- Add `Observable` variable type for substitution-based recording without adding variables to the DAE system.
- Add two-pass initialization with block-to-Observable conversions and discrete dependency tracking.
- Add pre-evaluation of discrete flags before the initialization pass.
- Add nonmonotone backtracking line search to the Newton-Raphson solver in both PFlow and TDS for improved convergence robustness.
- Separate `j_numeric` into two hooks: `j_setup` (one-time sparsity pattern and constant Jacobian registration, called by `store_sparse_pattern`) and `j_numeric` (per-iteration numerical Jacobian update, called by `j_update`, parallel to `g_numeric`/`f_numeric`). Add `flags.j_setup` for the former; `flags.j_num` now controls the per-iteration hook.

Model changes:

- Add `SHAFT5` 5-mass torsional shaft model (HP-IP-LP-Rotor-EX) for sub-synchronous resonance studies on synchronous generators. Includes `DynShaft` group and IEEE 14-bus test case (`ieee14_shaft5.json`).
- Add `REECB1` renewable energy control model with test cases.
- Remove `ug` ExtParam and custom `ue` ConstService from `ExcBase` and `TGBase`; now inherited from the base `Model` class.
- Remove `uee` ExtService and custom `ue` ConstService from `PSSBase`.
- Rename DGPrct `ue` Algeb (lock flag) to `lock` to avoid collision with the new base `ue`.
- Fix EXDC2 `Se0` formula to use `ue` instead of removed `ug`.
- Enable `AntiWindup` init limit adjustment and fix REEC models.
- Fix `zero_out` bypass in `LeadLag` and `LeadLag2ndOrd` blocks.
- Add `FreqDiv` frequency divider model (Milano & Ortega 2017) for algebraic bus frequency estimation from generator rotor speeds and network susceptance. Uses constant Jacobians via `j_setup` for zero per-iteration overhead.

Eigenvalue analysis:

- Replace permutation-based reorder with a fold/eliminate/reduce pipeline in eigenvalue analysis for improved numerical robustness.

State estimation:

- Add LAV (Least Absolute Value) estimator via iteratively reweighted least squares (IRLS), robust to gross measurement errors. Pass `algorithm=lav` to `SE.run()`.
- Vectorize weight handling in both WLS and LAV to avoid dense diagonal matrix construction.
- Add chi-squared test guard that rejects with a warning for non-WLS algorithms.
- Raise default `max_iter` from 20 to 100 for IRLS convergence.

Time-domain simulation:

- Add QNDF variable-order (1–5) quasi-constant-step NDF integration method (Shampine & Reichelt 1997, equivalent to MATLAB `ode15s` / Julia `QNDF()`). Select with `ss.TDS.config.method = 'qndf'`. Features error-based adaptive step control with `abstol`/`reltol` tolerances, automatic order selection, D-table rescaling, and event-aware cache reset.
- Add `trap_adapt` adaptive trapezoidal integration method with step-doubling Richardson extrapolation for local truncation error (LTE) estimation. Select with `ss.TDS.config.method = 'trap_adapt'`. Features `abstol`/`reltol` error control, no-LTE recovery mode near events, and automatic `fixt=0` override.
- Add `dtmax` TDS config parameter to explicitly cap the maximum step size for variable-step methods (`0` = auto from frequency and time span).
- Refactor step-size control into polymorphic `calc_h()` on each integration method class, replacing the monolithic if/elif/else chain in `TDS.calc_h`. Extract shared niter-based heuristic into `ImplicitIter.niter_next_h()` and shared adaptive bust-check into `check_adaptive_bust()`.
- Add `requires_variable_step` class attribute to integration methods, replacing the removed `adaptive` flag.
- Add `TDS.reinit()` for fast idempotent reset to the post-init state, enabling RL training loops with ~1ms per reset instead of full reload. Restores DAE state `(x, y)`, device status `(u, ue)`, ConstService values, and NumParam values, then re-derives discrete flags and residuals via `fg_update(init=True)`. Internally, `Model.snapshot_init()` / `Model.restore_init()` save and restore model-level mutable state; `NumParam` and `ConstService` each store a `_v_t0` snapshot.

Reinforcement learning:

- Add `andes.rl` module with `AndesEnv`, a Gymnasium-compatible environment for reinforcement learning on power system dynamics. Supports configurable observations (with optional device `idx` selection), group-level and model-level action setpoints, custom reward and disturbance functions, and fast reset via `TDS.reinit()`. Install with `pip install andes[rl]`.
- Add tutorial notebook (Tutorial 15: Reinforcement Learning) covering environment construction, observation and action specification, reward function design, disturbance injection, policy comparison, and integration with Stable-Baselines3.

PSS/E parser:

- Refactor RAW parser to a data-driven architecture with PSS/E v34 support.
- Fix DYR parser to handle embedded quoted strings and comma-separated values.

Refactoring:

- Refactor `System` internals by extracting `RegistryLoader` and `CodegenManager`; prefer direct calls via `system.registry.*` and `system.codegen.*`, with legacy `System` delegate methods deprecated for removal in v3.0.
- Convert the `System` module into a package and extract `DAECompactor`.
- Extract config runtime logic into `SystemConfigRuntime` (`andes.system.config_runtime`); prefer `system.config_runtime.*` over facade-level helpers. Legacy wrappers in `andes.system.facade` are deprecated and scheduled for removal in v3.0.
- Extract model evaluator logic into `ModelEvaluator` (`andes.core.model.model_evaluator`) using composition. Keep `Model` methods (`f_update`, `g_update`, `j_update`, service/discrete updates, and input refreshers) as delegate wrappers to preserve call sites.
- Move connectivity check into `ConnMan` and optimize.
- Replace `ConnMan` bus propagation with declarative status framework.
- Refactor `SystemConfigRuntime` into phased config resolution (`load_rc` → `merge_file_config` → `apply_cli_overrides` → `finalize`) and restructure `System.__init__` with `_init_managers` / `_init_models` helpers.
- Split `System.config` into `system.config` (case-relevant: `freq`, `mva`, `diag_eps`, `warn_abnormal`) and `system.runtime` (machine/environment: `numba`, `sparselib`, `dime_*`, `np_*`, etc.). Only case config is written to data files; runtime config persists in `andes.rc` under `[Runtime]`.
- Move per-routine `sparselib` to system-wide `system.runtime.sparselib`.
- Add `Config.check()` warning for unrecognized config fields (typos, misplaced keys) and `Config._add()` warning for fields moved to `[Runtime]` in v2.0.

I/O:

- Add embedded configuration support: xlsx files gain a `_config` sheet and JSON files gain a `_config` key, both using a `section`/`key`/`value` record format. File-embedded config overrides `andes.rc` defaults but is itself overridden by CLI `config_option`.
- Route `u` changes through `set_status` and remove `Bus.set` override.
- Replace `ModelCache` boilerplate with declarative registration.
- Rewrite TDS init error diagnostics with limit clamping report and DAE reverse map.

I/O and file loading:

- Extend `--addfile` (`-a`) to accept multiple files and support cross-format loading. Any supported format (xlsx, json, matpower) can be used as an additional file on top of any base case format. The `-a` flag can be repeated on the command line (e.g., `-a f1.dyr -a f2.xlsx`), and the Python API accepts both a string and a list.

API improvements:

- Consolidate `set()` and `alter()` into a unified `set()` method with a cleaner positional signature: `set(src, idx, value, base=None)`. The old `alter()` method is deprecated and will be removed in v3.0. See the upgrade guide below.
- Add `TDS.get_timeseries(var)` convenience method that returns time-series data as a pandas DataFrame, dispatching automatically by variable type (State, Algeb, ExtState, ExtAlgeb, Observable).
- Add fuzzy-match warnings for unrecognized field names in `System.add()`, suggesting close matches with `difflib.get_close_matches`.
- Rename `System.add()` first positional parameter from `model` to `model_name` to resolve the keyword collision with `Alter` and `Toggle`'s `model` device parameter. All models (including `Alter` and `Toggle`) can now use kwargs: `ss.add('Toggle', model='Line', dev='Line_5', t=1.0)`. Dict form remains supported for backward compatibility.
- Add `CPF.run_qv()` and `CPF.plot_qv()` for QV curve analysis at a specified bus. Delegates to the existing continuation engine with Q-only targets; results stored in `qv_q`, `qv_v`, and `qv_bus`.
- Add Plotly backend for interactive TDS plots in Jupyter notebooks via `ss.TDS.plt.plot(..., backend='plotly')`. Supports zoom, pan, hover tooltips, horizontal/vertical reference lines, figure reuse, and HTML export. LaTeX legend labels are automatically replaced with plain text.

Packaging:

- Make `numba` an optional dependency. Install with `pip install andes[performance]` to enable Numba JIT compilation. ANDES runs without numba by default (`runtime.numba = 0`); if numba is enabled but not installed, a warning is logged and execution continues without JIT.

Upgrade guide — `set()` / `alter()` consolidation:

- **`alter()` → `set()` with `base='device'`**: `alter()` accepted values in device (input) base, converted them via `pu_coeff`, and updated the `vin` record. The new equivalent is `set(..., base='device')`.

```
# Before (deprecated, emits FutureWarning):
ss.GENROU.alter('M', 1, 10.0)

# After:
ss.GENROU.set('M', 1, 10.0, base='device')
```

- **`set()` old-style keyword arguments**: The old `set(src, idx, attr, value)` four-argument form is deprecated. Drop the `attr` argument (it defaults to `'v'`):

```
# Before (deprecated, emits FutureWarning):
ss.GENROU.set('M', 'GENROU_1', 'v', 2.0)

# After:
ss.GENROU.set('M', 'GENROU_1', 2.0)
```

- **Default base is system base**: `set(src, idx, value)` writes the value directly in system per-unit, matching what `get()` returns. This makes read-modify-write round-trips correct without conversion:

```
h = ss.GENROU.get('M', 'GENROU_1')    # system-base value
ss.GENROU.set('M', 'GENROU_1', h * 1.2)    # 20% increase, same base
```

- **Time constant updates are automatic**: When modifying a time constant parameter (one whose variable has a `t_const` attribute), `set()` automatically updates `dae.Tf` and `TDS.Teye`.

Logging:

- Unify all runtime output through Python's `logging` module with a single `TqdmStreamHandler` that delegates to `tqdm.write()` to avoid corrupting progress bars. Removes 9 ad-hoc `tqdm.write()` call sites and their `system.options["verbose"]` guards. Timer events (Toggle, Fault, Alter, TimeSeries) log at INFO; convergence diagnostics (daeint, QNDF) log at DEBUG. Verbosity is now controlled entirely by `logging.setLevel()` (CLI: `andes -v 10` for DEBUG, `-v 30` for WARNING-only). Library callers (e.g., `andes.rl`) are silent by default since no handler is installed without an explicit `config_logger()` call.
- Convert operational `print()` calls in `codegen.py` and `io/txt.py` to `logger.info()` / `logger.error()`. CLI user-facing output (`print_license`, `versioninfo`, `plot`) remains as `print()`.

Bug fixes:

- TDS no longer checks connectivity at initialization time.
- Fix `pflow_tds` not updated during TDS initialization.
- Guard `save_unconstrained` against non-numeric `u.v` in discrete components.
- Disable discrete warning when initialized value is exactly at the limit.
- Clarify error message for device lookup failures in `find_idx`.
- Show slack generator count and bus indices in connectivity warning.
- Pin pandas version for pandapower compatibility.

5.6.2 v1.10 Notes

v1.10.1 (2026-03-09)

- Fix `zero_out` bypass in `LeadLag` and `LeadLag2ndOrd` blocks. `LeadLag` now checks only `T2` (not both `T1` and `T2`) to activate bypass. `LeadLag2ndOrd` also checks only `T2` instead of requiring all four time constants to be zero.
- Replace permutation-based eigenvalue reorder with a fold/eliminate/reduce pipeline in `EIG` routine. Fixes singular matrix errors for cases with zero-`Tf` states and degenerate model equations.
- Fix PSS/E convention for tap ratios in `_parse_transf_v33`; zero taps default to 1.0 pu.
- Suppress `tqdm` output when logging level is higher than INFO.
- Batch parameter correction warnings during setup. Instead of logging one warning per device, corrections (non-zero, non-positive, non-negative) are now grouped and reported once per violation type after all devices are added.

v1.10.0 (2026-01-05)

This release modernizes the build system and adds Pandapower 3 compatibility. The release also includes a major documentation overhaul.

Build system changes:

- Migrate to `pyproject.toml` and remove `setup.py` support.
- Remove Azure Pipelines configuration; CI is now handled entirely by GitHub Actions.
- Remove `versioneer.py` in favor of modern versioning.

Pandapower compatibility:

- Update pandapower interface to version 3.x compatibility.
- Fix interop and tests to be compatible with Pandapower 3.

PSS/E parser improvements:

- Refactor PSS/E data parsing functions to improve readability and maintainability.
- Enhance PSS/E RAW parser to support 3-winding transformers with `CZ=3` and `CM=2`. Implement conversion of load loss and impedance to R and X, and no-load loss and excitation current to G and B.
- Enhance PSS/E RAW parser to correctly handle quoted strings with commas and slashes.
- Correct a typo in the PSS/E DYR parser, changing "Tprod" to "Tpord".

Bug fixes:

- Fix `PVD1` frequency deadband to only respond to downward frequency deviations. Contributed by @J-K-Peng.
- Fix a bug in line model that causes incorrect admittance matrix.
- Fix NumPy NaN compatibility issue in `Model.set()`.
- In connectivity check, only call `SynGen.store_idx_island` when there is at least one SynGen.

Documentation improvements:

- Overhaul the documentation to use MyST and Sphinx 7.0.

Other changes:

- Enhance `Model.set()` to support assigning values when the destination is a list. Contributed by @jinningwang.
- Add `ipywidgets` to dependencies.
- Update dependencies in `pyproject.toml` for compatibility and version constraints.
- Add a demo `QSTS.ipynb` to show how to run QSTS simulation in ANDES. See folder `andes/examples/demonstration`. Contributed by @jinningwang.
- Update broken link to ESST1A exciter documentation. Contributed by @isilber.

5.6.3 v1.9 Notes

v1.9.3 (2025-01-05)

Development of connectivity manager `ConnMan`:

- Add case `ieee14_conn.xlsx` for demonstration.
- Add `ConnMan` class to manage connectivity.
- Add `ConnMan` to `System` to as an attribute `conn`.
- Add a demo notebook `ConnMan.ipynb`. See folder `andes/examples/demonstration`.

Other changes:

- In the `dae` module, change `self.t.itemset` to array assignment to ensure compatibility with NumPy 2.0.
- Follow RTD's deprecation of Sphinx context injection at build time
- In symbolic processor, most variables are assumed to be real, except some services that are specified as complex. This will allow generating simplified expressions.
- Adjust `BusFreq.Tw.default` to 0.1.
- Add parameter from_csv=None in TDS.run() to allow loading data from CSV files at TDS begining.
- Fix `TDS.init()` and `TDS._csv_step()` to fit loading from CSV when `Output` exists.
- Add function signature `allow_all=False` to `ModelData.find_idx()` `GroupBase.find_idx()` to allow searching all matches.
- Enhanced three-winding transformer parsing in PSS/E raw files by assigning the equivalent star bus `area`, `owner`, and `zone` using the high-voltage bus values.
- Add function signature `attr='v'` to method `Model.alter` for altering parameter values without manual per unit conversion
- Add following methods to `GroupBase` for easier usage: `get_all_idxes`, `alter`, `as_dict`, and `as_df`
- Add two demo `add_RenGen.ipynb` and `replace_SynGen.ipynb` to show how to do renewable penetration via code. See folder `andes/examples/demonstration`
- Add a demo `manipulate_params.ipynb` to compare the differnce between `set` and `alter`
- Extend `SynGen` common parameters with `bus`, `gen`, `Sn`, `Vn`, and `fn`
- Extend `StaticGen` common parameters with `bus`
- Fix `TDS.init()` and `TDS._csv_step()` to fit loading from CSV when `Output` exists.
- Add parameter `allow_all=False` to `ModelData.find_idx()` `GroupBase.find_idx()` to allow searching all matches.
- Add method `GroupBase.get_all_idxes()` to get all indices of a group.

- Enhanced three-winding transformer parsing in PSS/E raw files by assigning the equivalent star bus `area`, `owner`, and `zone` using the high-voltage bus values.
- Minor fix on model `TGOV1NDB`
- Clarification on models `ACEc` and `ACE` given NERC's definition, as NERC suggesst that both bias and beta should be nevative
- Minor fixes for finding `dvipng` executable.

v1.9.2 (2024-03-25)

- Improve PSS/E parser for the `wmod` field in the static generator section.
- Consider line status when parsing PSS/E file.
- Added functions in `Line` for building network admittance matrix, `Bdc` matrix for DC power flow, and `Bp` and `Bpp` matrices for fast decoupled power flow. See `build_y`, `build_b` and `build_Bdc`.

v1.9.1 (2024-02-04)

This is a hotfix to pass `pip check` for KVXOPT version.

v1.9.0 (2024-02-01)

- Initially, `dae.t` is set to `-1.0` during power flow calculation. TDS initialization will set `dae.t = 0.0`.
- Rewritten the islanding detection algorithm.
- In `system.py`, import `importlib.util` to workaround unexported `util` module in Python 3.10.

5.6.4 v1.8 Notes

v1.8.10 (2023-08-10)

- An internal change to the xlsx parser option. Previously, the index column was set to 0, which assumes the existence of `uid` as the header. Manually input sheets without `uid` will fail. This version unsets the index column when using pandas to parse the xlsx file.
- `Jumper` is included for connectivity check.
- Add curly brackets for LaTeX symbols from sub-blocks. Nested symbols with subscripts and superscripts now display properly.

v1.8.9 (2023-06-22)

- Fix `Model.alter()` for parameters that are time constants for differential equations. Altered time constants are now properly updated in `dae.Tf` and will be correctly reflected in simulation results.
- Fix a bug in connectivity check.
- Support numba for Python 3.11.

v1.8.8 (2023-04-24)

- Repository transferred to `github.com/curent/andes`.
- Fix for frequency-dependent load model `FLoad` by not exporting the external `bus` parameter.

v1.8.7 (2023-03-10)

- Added IEEE 39-bus test case with dynamics in the ANDES xlsx format. Contributed by @jinningwang.
- In `interop.pandapower`, the `_verify_pf` function will not be called if TDS has been initialized. This prevents the breaking of variable addresses.

v1.8.6 (2023-02-13)

- Minor fix in `timeseries.py` for backward compatibility with Python 3.6.

v1.8.5 (2022-12-22)

- Fixed an issue to properly handle turning off a `Jumper`.
- New interface to GridCal by @JosepFanals. GridCal installation required.
- Bug fixes in WTARA1.

v1.8.4 (2022-11-23)

- Fixed a bug in time stepping where step sizes were not properly reduced when the solver failed to converge.

v1.8.3 (2022-11-15)

- Support Python 3.11. The average performance gain is a few percent.
- Made `numba` an optional dependency because it does not yet support Python 3.11.
- Fixes a bug in the time stepping logic that causes resumed simulations to overwrite the result of the previous final step.

v1.8.2 (2022-10-30)

This is a minor release to support SymPy 1.11.x.

Internally, `ModelData` contains a new member field `index_bases` to record the variables by which other variables are indexed. Most variables, by default, are indexed by `idx`. Some models (such as COI) whose device can link to multiple other devices can have multiple index bases.

Index bases are currently placeholders and have not been used in the current version.

v1.8.1 (2022-09-24)

- New turbine governor model *HYGOV4*. Thanks to Zaid Mahmood for the contribution.
- Bug fixes to saving data to xlsx file.

v1.8.0 (2022-08-30)

- Internal change: drop the support for reloading generated code from `calls.pkl`. All generated code are serialized into Python files and reloaded as a `pycode` module.
- Fix an issue where cases in multiprocessing fail to serialize due to not being able to find `pycode`. `pycode` is now properly imported by the main process.
- When one needs to serialize a System object, such as during multiprocessing, one needs to manually import `pycode`. This can be done by calling `andes.system.import_pycode()`.
- Internal change: serializing with `dill` is not set to recursive by default.

5.6.5 v1.7 Notes

v1.7.8 (2022-08-24)

- Support marking tests as extra so that they are not run by default. The function names for extra tests should contain `extra_test`. To run all tests, use `andes st -e` or `andes selftest --extra`.
- Add the `new_A` flag for sparse solvers to trigger actions. Currently, only the `spsolve` solver will need to rebuild and refactorize the sparse matrix.

v1.7.7 (2022-08-02)

- Implemented a chattering detection and stop algorithm by increasing the step size when chattering is detected.
- `TimeSeries.get_data()` works for systems with `Output`.
- Allow freezing states associated with anti-windup limiters after a certain number of iterations to prevent chattering.

Models:

- New exciter models: ESDC1A, EXAC2.
- New grid-forming inverter models: REGF1, REGF2, REGF3.
- For `IEEEG1`, normalize `K1` through `K8` if they do not sum up to 1.0

v1.7.6 (2022-07-11)

- In eigenvalue analysis, added parameter sweeping and scatter plot for root loci. See revised Example 4.
- Documentation improvements.

v1.7.5 (2022-07-05)

- Added *Fortescue* model to support symmetric component calculation. The model allows interfacing a positive-sequence bus with three buses representing three phases. See the model description for details.
- Added *PLL2* for the Synchronously-rotating Reference Frame (SRF)-based PLL.
- *REGCP1* works identically to *REGCA1* when the `pll` parameter is empty and works with a PLL.
- *REECA1* is updated to work with `vd` of the converter. If using the `REGCA1` model, `vd = v`.
- Reverted a change in Line parameter that caused SMIB case to crash.

v1.7.4 (2022-07-01)

- Renamed model `Toggler` to `Toggle`.
- New model: ESAC5A. (Contributed by Ahmad Ali).
- Added documentation for creating disturbances.
- Updated documentation for modeling blocks.

v1.7.3 (2022-06-25)

Bug fix:

- Fix `Ipcmd` initialization equation of `REGCA1`.

Improved the interface to pandapower:

- Improved `to_pandapower` performance using vectorized conversion.
- Enhanced `make_link_table` to include group `RenGen`.

v1.7.2 (2022-06-07)

- Improved documentation and examples.

v1.7.1 (2022-05-31)

This release contains minor fixes to the documentation.

Other changes:

- In `PVD1`, Enable `pmx` limiting by default.
- In `ST2CUT`, fix the type of `busr2` and `busr` to `IdxParam`.

v1.7.0 (2022-05-22)

Allow incrementally offloading simulation data from memory to the output file:

- `[TDS].limit_store` is a boolean value to enable the limit for in-memory time-series storage. If set to `1`, data will be offloaded to the `npz` file every `[TDS].max_store` steps. Offloaded data will then be erased from memory.
- If you need to interact with the time-series data in memory, you need to keep `[TDS].limit_store` to `0`.

Allow specifying models, variables, and/or devices to output:

- See *Output*. The `model` field is mandatory. Leaving `varname` or `dev` blank indicates the selection of all applicable elements. For example, specifying `model` and `varname` without `dev` means that the variable for all devices will be exported.
- Plot tool works with in-memory time-series data specified by Output.

Simulation output control:

- Allow controlling the save frequency for output data in `[TDS].save_every`. The default value is `1`, which means that every step will be saved. Setting it to `4`, for example, will save data every four steps. This setting applies to the in-memory storage and the output data file.
- Setting `save_every = 0` will immediately discard all data after each simulation step.

- Added the option `[TDS].save_mode` to change the automatic simulation data dumping to manual. Accepted values are `auto` and `manual`. This option shall only be adjusted to `manual` when one is manually stepping the simulation and wants to avoid writing to the output file when the simulation reaches `TDS.config.tf`. One will need to call `TDS.save_output()` when the full simulation concludes to avoid losing unsaved data.

Other changes:

- Fix the initialization of offline synchronous generators.
- Allow styles to be set for plots using the argument `style`. To generate figures for IEEE publications, use `style=ieee` (require package `scienceplots`).
- Moved the writing of the `lst` file to the first step of simulation.
- `andes misc -C` will not remove `_out.csv` file as it is considered data for post-processing just like exported figures.

5.6.6 v1.6 Notes

v1.6.6 (2022-04-30)

- Rename `[System] call_stats` to `[System] save_stats` for clarity. If turned on, one can retrieve statistics of function calls in `TDS.call_stats`.
- Store routine execution time to routine member `execution_time`.
- Fix PSS/E parsing issues with *GAST*.
- Fix issues and update default parameters for `REGCV1` and `REGCV2`.
- Allow adjusting limits for state variables during initialization. Like for algebraic variables, the default setting automatically adjusts the upper limit but not the lower one.

v1.6.5 (2022-04-19)

- Added a TDS stop criteria based on rotor angle separation with reference to Power System Analysis Toolbox.
- Fix a bug for snapshot save and load. It now supports writing to and reading from `io.BytesIO()`.

v1.6.4 (2022-04-17)

Breaking change:

- `PV` model no longer has `p` as a variable in the DAE. `p` copies the value of `p0`. This change affects the addresses of variables.
- Changed `models.file_classes` to a list to improve the control over the class initialization sequence in the same package.

Operator splitting for internal algebraic variables:

- `VarService` can be evaluated model-internal algebraic variables outside the DAE system. This approach is known as operator splitting and is commonly used in other simulation tools.
- Operator splitting reduces the size of the DAE system but introduces a one-iteration lag between the internal algebraic variables and others in the DAE system.
- `VarService` shall be avoided for singular functions (non-continuous) and shall not be adopted to circumvent initializing algebraic equations.
- `VarService` takes an argument `sequential`, which is `True` by default. Non-sequential `VarService` shall not depend on other `VarService` calculated at the same step as they will be evaluated simultaneously.
- *`andes.interop.pandapower.to_pandapower()`* set all generators as controllable by default. Generators in converted the pandapower case are named using the `idx` of `StaticGen`.
- Bug fixes in `interop.pandapower.make_link_table()`.

Other changes:

- Added a new service type `andes.core.service.SubsService` for temporary symbols that will be substituted at code generation time.
- `TDS.plt.plot()` now accepts a list of variable objects. For example, `ss.TDS.plt.plot([ss.GENROU.omega, ss.GENROU.delta], a=[0, 1])` will plot the rotor speed and angles of the 0-th and the 1-st generator.
- Added *REGCP1* model for generic converters with PLL support.
- Fixed PSS/E parser for *HYGOV*.

v1.6.3 (2022-04-06)

- Adjustments in the Pandapower interface. Added `make_GSF()` for the generation shift factor matrix.
- Reduced import overhead for the command-line tool.

v1.6.2 (2022-03-27)

Interoperability:

- Added interoperability modules for MATPOWER (through Oct2Py), pandapower and pypowsybl.
- Added Examples and API reference for the interoperability module.
- Improved the setup script to support extra dependencies. The following extras groups are supported: `dev` and `interop`.
- Added tests for power flow calculation against MATPOWER.

Others:

- Added a shorthand command `andes.system.example()` to return a disposable system. It can be useful for quick prototyping.

- Improved the formatting and navigation of Model references.
- Models store the base values for per-unit conversion in `Model.bases`.

v1.6.1 (2022-03-13)

- Revamped documentation with a much improved "Getting started" section.

v1.6.0 (2022-03-11)

- Migrated documentation to the pydata template.
- Added compatibility with SymPy 1.9 and 1.10.

5.6.7 v1.5 Notes

v1.5.12 (2022-03-05)

- Improved PSS/E parsers for WTDTA1 model to follow PSS/E parameter definition.
- Included the Jupyter notebook examples in the documentation.
- Tweaks to the plot utility.

v1.5.11 (2022-02-23)

- Reduced the tolerance for tiny variable increments to be treated as zero.
- Fixed PSS/E parsers for renewable models.
- Minor renewable model fixes.

v1.5.10 (2022-02-01)

- Fixed one equation in `REGC_A`.

v1.5.9 (2022-01-31)

- Added PLL1, a simple PLL model.
- Renamed `REGCVSG` to `REGCV1` and `REGCVSG2` to `REGCV2`.
- Added an alias list for model names. See `models/__init__.py`.
- Multiprocessing now executes on all CPUs that are physical, instead of logical. A new package `psutil` needs to be installed.
- Use of `Selector` is deprecated.

v1.5.8 (2021-12-21)

- Full initialization debug message will be printed only when `-v 10` and `run --init` are both used.
- Improved warning of out-of-limit initialization. Variables initialized at limits will be shown only at the debug level.
- Initialization improvements for models `REGCA1` and `REECA1`.
- Added model `HYGOV`.
- Changed the default `vout` of offline exciters to zeros. All `vout` equations need to be multipled by `ue`.

v1.5.7 (2021-12-11)

This minor release highlights the improved debugging of initialization.

Highly verbose initialization output can be enabled when the verbose level is 10 or less. For example,

```
andes -v 10 run test.xlsx -r tds --init
```

will set the verbose level to 10 and run `test.xlsx` in the current folder, proceed to time-domain simulation but only initialize the models. Outputs will be printed to the shell where the command is executed.

To save the output to a file, use the following in a UNIX shell:

```
andes -v 10 run test.xlsx -r tds --init > info.txt 2>&1
```

where the first > pipes the output to a file named `info.txt`, and `2>&1` appends stderr (2) to stdout (1).

The other main improvement is allowing automatic limit adjustment during initialization. Due to parameter errors, some variables will be initialized to values outside the given limits. Most commercial software does not attempt to fix the parameter but rather adjust the limit in run time.

The same approach is followed in ANDES by automatically adjusting the upper limit, if exceeded, to variable initial values. The lower limit, however, is kept unadjusted by default.

Discrete components now take an argument named `allow_adjust` so that the model developer can specify if its limits can be adjusted or must be kept as is. Each model is allowed to specify three config flags to customize runtime behaviors: `allow_adjust`, `adjust_lower`, and `adjust_uppwer`. By default, `allow_adjust=True`, `adjust_upper=True`, and `adjust_lower=False`. One can modify the config file to enable or disable the limit adjustments for specific models.

Other fixes include:

- Bug fixes for `GAST` parameter `AT`.
- Bug fixes for `IEEET3`, `GAST`, `ESAC1A` and `ESST1A` when device is off to avoid matrix singularity.

v1.5.6 (2021-11-25)

- Allow specifying config options through command-line arguments `--config-option`.
- Added a voltage and frequency playback model `PLBVFU1`.
- Bug fixes to an SEXS equation.

v1.5.5 (2021-11-13)

- Added a `Timeseries` model for reading timeseries data from xlsx.
- Converted several models into Python packages.
- Bug fixes to TGOV1 equations (#226)

v1.5.4 (2021-11-02)

- Fixed a bug in generated `select` functions that omitted the coefficients of `__ones`.

v1.5.3 (2021-10-31)

- Revered special arguments for the generated `select` function.
- Stabilized the argument list of pycode. If the pycode is identical to existing ones, the existing file will not be overwritten. As a result, compiled code is fully cached.
- Partially seperated time-domain integration method into `daeint.py`.

v1.5.2 (2021-10-27)

- Removed `CVXOPT` dependency.
- Removed `__zeros` and `__ones` as they are no longer needed.
- Added `andes prep -c` to precompile the generated code.
- Added utility functions for saving and loading system snapshots. See `andes/utils/snapshot.py`.
- Compiled numba code is always cached.
- Bug fixes.

v1.5.1 (2021-10-23)

- Restored compatibility with SymPy 1.6.
- Added a group for voltage compensators.
- New models: `IEEEVC` and `GAST`.

v1.5.0 (2021-10-13)

- Support numba just-in-time compilation of all equation and Jacobian calls.

This option accelerates simulations by up to 30%. The acceleration is visible in medium-scale systems with multiple models. Such systems involve heavy function calls but a rather moderate load for linear equation solvers. The speed up is less significant in large-scale systems where solving equations is the major time consumer.

Numba is required and can be installed with `pip install numba` or `conda install numba`.

To turn on numba for ANDES, in the ANDES configuration under `[System]`, set `numba = 1` and `numba_cache = 1`.

The just-in-time compilation will compile the code upon the first execution based on the input types. When compilation is triggered, ANDES may appear frozen due to the compilation lag. The option `numba_cache = 1` will cache compiled machine code, so that the lag only occurs once until the next `andes prep`.

- Allow `BackRef` to populate to models through `Group`.

When model `A` stores an `IdxParam` pointing to a group, if `BackRef` with the name `A` are declared in both the group and the model, both `BackRef` will retrieve the backward references from model `A`.

- Allow `BaseVar` to accept partial initializations.

If `BaseVar.v_str_add = True`, the value of `v_str` will be added in place to variable value. An example is that voltage compensator sets part of the input voltage, and exciter reads the bus voltage. Exciter has `v.v_str_add = True` so that when compensators exist, the input voltage will be bus voltage (vbus) plus (Eterm - vbus). If no compensator exists, exciter will use bus voltages and function as expected.

- Added reserved variable names `__ones` and `__zeros` for ones and zeros with length equal to the device number.

`__ones` and `__zeros` are useful for vectorizing `choicelist` in `Piecewise` functions.

5.6.8 v1.4 Notes

v1.4.4 (2021-10-05)

- Bug fixes for refreshing generated code.

v1.4.3 (2021-09-25)

This release features parallel processing that cuts the time for `andes prepare` by more than half.

- `andes prepare` supports multiprocessing and uses it by default.
- Added aliases `andes st` and `andes prep` for `andes selftest` and `andes prepare`.
- `andes.config_logger` supports setting new `stream_level` and `file_level`.

New exciter models are contributed by Jinning Wang.

- Added `AC8B`, `IEEET3` and `ESAC1A`.

Other changes include disallowing numba's `nopython` mode.

v1.4.2 (2021-09-12)

- Bug fixes
- Dropped support for `cvxoptklu`.

v1.4.1 (2021-09-12)

- Bug fixes.
- Overhaul of the `prepare` and `undill` methods.
- `andes prepare` can be called for specific models through `-m`, which takes one or many model names as arguments.

v1.4.0 (2021-09-08)

This release highlights the distributed energy resource protection model.

- Added `DGPRCT1` model to provide DG models with voltage- and frequency-based protection following IEEE 1547-2018.
- `REECA1E` supports frequency droop on power.
- Throws TypeError if type mismatches when using ExtAlgeb and ExtState.

5.6.9 v1.3 Notes

v1.3.12 (2021-08-22)

Plot enhancements:

- `plot()` takes an argument `mark` for masking y-axis data based on the `left` and `right` range parameters.

- `TDS.plt` provides a `panoview` method for plotting an panoramic view for selected variables and devices of a model.

Models:

- Added WIP EV models and protection models.

Test case: - Added CURENT EI test system. - Added a number of IEEE 14 bus test systems for specific models.

v1.3.11 (2021-07-27)

- Added `REECA1E` model with inertia emulation.
- Fixed an issue where the `vtype` of services was ignored.
- Changed default DPI for plotting to 100.

v1.3.10 (2021-06-08)

- Bug fixes for controllers when generators are off.

v1.3.9 (2021-06-02)

- Bug fixes in exciters when generators are offline.
- Added `safe_div` function for initialization equations.

v1.3.8 (2021-06-02)

- Added `REGCVSG` model for voltage-source controlled renewables.
- Turbine governors are now aware of the generator connection status.

v1.3.7 (2021-05-03)

- Allow manually specifying variables needing initialization preceding a variable. Specify a list of variable names through `BaseVar.deps`.

v1.3.6 (2021-04-23)

- Patched ESD1 model. Converted `distributed.py` into a package.
- Bug fixes.

v1.3.5 (2021-03-20)

- Fixed a bug in connectivity check when bus 0 is islanded.
- Updated notebook examples.
- Updated tutorials.

v1.3.4 (2021-03-13)

- Fixed a bug for the generated renewable energy code.

v1.3.2 (2021-03-08)

- Relaxed the version requirements for NumPy and SymPy.

v1.3.1 (2021-03-07)

- Writes all generated Python code to `~/.andes/pycode` by default.
- Uses generated Python code by default instead of `calls.pkl`.
- Works with NumPy 1.20; works on Apple Silicon (use `miniforge`) to install native Python and NumPy for Apple Silicon.
- Generalized model initialization: automatically determines the initialization sequence and solve equations iteratively when necessary.
- In `System.config`, `save_pycode` and `use_pycode` are now deprecated.

v1.3.0 (2021-02-20)

- Allow `State` variable set `check_init=False` to skip initialization test. One use case is for integrators with non-zero inputs (such as state-of-charge integration).
- Solves power flow for systems with multiple areas, each with one Slack generator.
- Added `Jumper` for connecting two buses with zero impedance.
- `REGCA1` and synchronous generators can take power ratio parameters `gammap` and `gammaq`.
- New models: `IEESGO` and `IEEET1`, `EXAC4`.
- Refactored exciters, turbine governors, and renewable models into modules.

5.6.10 v1.2 Notes

v1.2.9 (2021-01-16)

- Added system connectivity check for islanded buses.
- Depend on `openpyxl` for reading excel files since `xlrd` dropped support for any format but `xlsx` since v2.0.0.

v1.2.7 (2020-12-08)

- Time-domain integration now evaluates anti-windup limiter before algebraic residuals. It assures that algebraic residuals are calculated with the new state values if pegged at limits.
- Fixed the conditions for Iq ramping in REGC; removed `Iqmax` and `Iqmin`.
- Added a new plot function `plotn` to allow multiple subplots in one figure.
- `TDS.config.g_scale` is now now used as a factor for scaling algebraic equations for better convergence. Setting it to 1.0 functions the same as before.

v1.2.6 (2020-12-01)

- Added `TGOV1N` model which sums `pref` and `paux` after the 1/droop block.
- Added `ZIP` and `FLoad` for dynamic analysis. Need to be initialized after power flow.
- Added `DAETimeSeries.get_data()` method.
- Added IEEE 14-bus test cases with solar PV (ieee14_solar.xlsx) and Generic Type 3 wind (ieee14_wt3.xlsx).

v1.2.5 (2020-11-19)

- Added `Summary` model to allow arbitrary information for a test case. Works in `xlsx` and `json` formats.
- PV reactive power limit works. Automatically determines the number of PVs to convert if `npv2pq=0`.
- Limiter and AntiWindup limiter can use `sign_upper=-1` and `sign_lower=-1` to negate the provided limits.
- Improved error messages for inconsistent data.
- `DAETimeSeries` functions refactored.

v1.2.4 (2020-11-13)

- Added switched shunt class `ShuntSw`.
- BaseParam takes `inconvert` and `oconvert` for converting parameter elements from and to files.

v1.2.3 (2020-11-02)

- Support variable `sys_mva` (system base mva) in equation strings.
- Default support for KVXOPT through `pip` installation.

v1.2.2 (2020-11-01)

New Models:

- `PVD1` model, WECC distributed PV model. Supports multiple PVD1 devices on the same bus.
- Added `ACEc` model, ACE calculation with continuous freq.

Changes and fixes:

- Renamed `TDS._itm_step` to `TDS.itm_step` as a public API.
- Allow variable `sys_f` (system frequency) in equation strings.
- Fixed ACE equation. measurement.
- Support `kvxopt` as a drop-in replacement for `cvxopt` to bring KLU to Windows (and other platforms).
- Added `kvxopt` as a dependency for PyPI installation.

v1.2.1 (2020-10-11)

- Renamed `models.non_jit` to `models.file_classes`.
- Removed `models/jit.py` as models have to be loaded and instantiated anyway before undill.
- Skip generating empty equation calls.

v1.2.0 (2020-10-10)

This version contains major refactor for speed improvement.

- Refactored Jacobian calls generation so that for each model, one call is generated for each Jacobian type.
- Refactored Service equation generation so that the exact arguments are passed.

Also contains an experimental Python code dump function.

- Controlled in `System.config`, one can turn on `save_pycode` to dump equation and Jacobian calls to `~/.andes/pycode`. Requires one call to `andes prepare`.

- The Python code dump can be reformatted with `yapf` through the config option `yapf_pycode`. Requires separate installation.
- The dumped Python code can be used for subsequent simulations through the config option `use_pycode`.

5.6.11 v1.1 Notes

v1.1.5 (2020-10-08)

- Allow plotting to existing axes with the same plot API.
- Added TGOV1DB model (TGOV1 with an input dead-band).
- Added an experimental numba support.
- Patched `LazyImport` for a snappier command-line interface.
- `andes selftest -q` now skips code generation.

v1.1.4 (2020-09-22)

- Support `BackRef` for groups.
- Added CLI `--pool` to use `multiprocess.Pool` for multiple cases. When combined with `--shell`, `--pool` returns `System` Objects in the list `system`.
- Fixed bugs and improved manual.

v1.1.3 (2020-09-05)

- Improved documentation.
- Minor bug fixes.

v1.1.2 (2020-09-03)

- Patched time-domain for continuing simulation.

v1.1.1 (2020-09-02)

- Added back quasi-real-time speed control through `--qrt` and `--kqrt KQRT`.
- Patched the time-domain routine for the final step.

v1.1.0 (2020-09-01)

- Defaulted `BaseVar.diag_eps` to `System.Config.diag_eps`.
- Added option `TDS.config.g_scale` to allow for scaling the algebraic mismatch with step size.
- Added induction motor models `Motor3` and `Motor5` (PSAT models).
- Allow a PFlow-TDS model to skip TDS initialization by setting `ModelFlags.tds_init` to False.
- Added Motor models `Motor3` and `Motor5`.
- Imported `get_case` and `list_cases` to the root package level.
- Added test cases (Kundur's system) with wind.

Added Generic Type 3 wind turbine component models:

- Drive-train models `WTDTA1` (dual-mass model) and `WTDS` (single-mass model).
- Aerodynamic model `WTARA1`.
- Pitch controller model `WTPTA1`.
- Torque (a.k.a. Pref) model `WTTQA1`.

5.6.12 v1.0 Notes

v1.0.8 (2020-07-29)

New features and models:

- Added renewable energy models `REECA1` and `REPCA1`.
- Added service `EventFlag` which automatically calls events if its input changes.
- Added service `ExtendedEvent` which flags an extended event for a given time.
- Added service `ApplyFunc` to apply a numeric function. For the most cases where one would need `ApplyFunc`, consider using `ConstService` first.
- Allow `selftest -q` for quick selftest by skipping codegen.
- Improved time stepping logic and convergence tests.
- Updated examples.

Default behavior changes include:

- `andes prepare` now takes three mutually exclusive arguments, `full`, `quick` and `incremental`. The command-line now defaults to the quick mode. `andes.prepare()` still uses the full mode.
- `Model.s_update` now evaluates the generated and the user-provided calls in sequence for each service in order.
- Renamed model `REGCAU1` to `REGCA1`.

v1.0.7 (2020-07-18)

- Use in-place assignment when updating Jacobian values in Triplets.
- Patched a major but simple bug where the Jacobian refactorization flag is set to the wrong place.
- New models: PMU, REGCAU1 (tests pending).
- New blocks: DeadBand1, PIFreeze, PITrackAW, PITrackAWFreeze (tests pending), and LagFreeze (tests pending).
- `andes plot` supports dashed horizontal and vertical lines through `hline1`, `hline2`, `vline1` and `vline2`.
- Discrete: renamed `DeadBand` to `DeadBandRT` (deadband with return).
- Service: renamed `FlagNotNone` to `FlagValue` with an option to flip the flags.
- Other tweaks.

v1.0.6 (2020-07-08)

- Patched step size adjustment algorithm.
- Added Area Control Error (ACE) model.

v1.0.5 (2020-07-02)

- Minor bug fixes for service initialization.
- Added a wrapper to call TDS.fg_update to allow passing variables from caller.
- Added pre-event time to the switch_times.

v1.0.4 (2020-06-26)

- Implemented compressed NumPy format (npz) for time-domain simulation output data file.
- Implemented optional attribute `vtype` for specifying data type for Service.
- Patched COI speed initialization.
- Patched PSS/E parser for two-winding transformer winding and impedance modes.

v1.0.3 (2020-06-02)

- Patches `PQ` model equations where the "or" logic "|" is ignored in equation strings. To adjust PQ load in time domain simulation, refer to the note in `pq.py`.
- Allow `Model.alter` to update service values.

v1.0.2 (2020-06-01)

- Patches the conda-forge script to use SymPy < 1.6. After SymPy version 1.5.1, comparison operations cannot be sympified. Pip installations are not affected.

v1.0.1 (2020-05-27)

- Generate one lambda function for each of f and g, instead of generating one for each single f/g equation. Requires to run `andes prepare` after updating.

v1.0.0 (2020-05-25)

This release is going to be tagged as v0.9.5 and later tagged as v1.0.0.

- Added verification results using IEEE 14-bus, NPCC, and WECC systems under folder `examples`.
- Patches GENROU and EXDC2 models.
- Updated test cases for WECC, NPCC IEEE 14-bus.
- Documentation improvements.
- Various tweaks.

5.6.13 Pre-v1.0.0

v0.9.4 (2020-05-20)

- Added exciter models EXST1, ESST3A, ESDC2A, SEXS, and IEEEX1, turbine governor model IEEEG1 (dual-machine support), and stabilizer model ST2CUT.
- Added blocks HVGate and LVGate with a work-around for sympy.maximum/ minimum.
- Added services `PostInitService` (for storing initialized values), and `VarService` (variable services that get updated) after limiters and before equations).
- Added service `InitChecker` for checking initialization values against typical values. Warnings will be issued when out of bound or equality/ inequality conditions are not met.
- Allow internal variables to be associated with a discrete component which will be updated before initialization (through `BaseVar.discrete`).

- Allow turbine governors to specify an optional `Tn` (turbine rating). If not provided, turbine rating will fall back to `Sn` (generator rating).
- Renamed `OptionalSelect` to `DataSelect`; Added `NumSelect`, the array-based version of `DataSelect`.
- Allow to regenerate code for updated models through `andes prepare -qi`.
- Various patches to allow zeroing out time constants in transfer functions.

v0.9.3 (2020-05-05)

This version contains bug fixes and performance tweaks.

- Fixed an `AntiWindup` issue that causes variables to stuck at limits.
- Allow `TDS.run()` to resume from a stopped simulation and run to the new end time in `TDS.config.tf`.
- Improved TDS data dump speed by not constructing DataFrame by default.
- Added tests for `kundur_full.xlsx` and `kundur_aw.xlsx` to ensure results are the same as known values.
- Other bug fixes.

v0.9.1 (2020-05-02)

This version accelerates computations by about 35%.

- Models with flag `collate=False`, which is the new default, will slice DAE arrays for all internal vars to reduce copying back and forth.
- The change above greatly reduced computation time. For `kundur_ieeest.xlsx`, simulation time is down from 2.50 sec to 1.64 sec.
- The side-effects include a change in variable ordering in output lst file. It also eliminated the feasibility of evaluating model equations in parallel, which has not been implemented and does not seem promising in Python.
- Separated symbolic processor and documentation generator from Model into `SymProcessor` and `Documenter` classes.
- `andes prepare` now shows progress in the console.
- Store exit code in `System.exit_code` and returns to system when called from CLI.
- Refactored the solver interface.
- Patched Config.check for routines.
- SciPy Newton-Krylov power flow solver is no longer supported.
- Patched a bug in v0.9.0 related to `dae.Tf`.

v0.8.8 (2020-04-28)

This update contains a quick but significant fix to boost the simulation speed by avoiding calls to empty user-defined numerical calls.

- In `Model.flags` and `Block.flags`, added `f_num`, `g_num` and `j_num` to indicate if user-defined numerical calls exist.
- In `Model.f_update`, `Model.g_update` and `Model.j_update`, check the above flags to avoid unnecessary calls to empty numeric functions.
- For the `kundur_ieeest.xlsx` case, simulation time was reduced from 3.5s to 2.7s.

v0.8.7 (2020-04-28)

- Changed `RefParam` to a service type called `BackRef`.
- Added `DeviceFinder`, a service type to find device idx when not provided. `DeviceFinder` will also automatically add devices if not found.
- Added `OptionalSelect`, a service type to select optional parameters if provided and select fallback ones otherwise.
- Added discrete types `Derivative`, `Delay`, and `Average`,
- Implemented full IEEEST stabilizer.
- Implemented COI for generator speed and angle measurement.

v0.8.6 (2020-04-21)

This release contains important documentation fixes and two new blocks.

- Fixed documentations in `andes doc` to address a misplacement of symbols and equations.
- Converted all blocks to the division-free formulation (with `dae.zf` renamed to `dae.Tf`).
- Fixed equation errors in the block documentation.
- Implemented two new blocks: Lag2ndOrd and LeadLag2ndOrd.
- Added a prototype for IEEEST stabilizer with some fixes needed.

v0.8.5 (2020-04-17)

- Converted the differential equations to the form of `T \dot{x} = f(x, y)`, where T is supplied to `t_const` of `State/ExtState`.
- Added the support for Config fields in documentation (in `andes doc` and on readthedocs).
- Added Config consistency checking.
- Converted `Model.idx` from a list to `DataParam`.

- Renamed the API of routines (summary, init, run, report).
- Automatically generated indices now start at 1 (i.e., "GENCLS_1" is the first GENCLS device).
- Added test cases for WECC system. The model with classical generators is verified against TSAT.
- Minor features: `andes -v 1` for debug output with levels and line numbers.

v0.8.4 (2020-04-07)

- Added support for JSON case files. Convert existing case file to JSON with `--convert json`.
- Added support for PSS/E dyr files, loadable with `-addfile ADDFILE`.
- Added `andes plot --xargs` for searching variable name and plotting. See example 6.
- Various bug fixes: Fault power injection fix;

v0.8.3 (2020-03-25)

- Improved storage for Jacobian triplets (see `andes.core.triplet.JacTriplet`).
- On-the-fly parameter alteration for power flow calculations (`Model.alter` method).
- Exported frequently used functions to the root package (`andes.config_logger`, `andes.run`, `andes.prepare` and `andes.load`).
- Return a list of System objects when multiprocessing in an interactive environment.
- Exported classes to `andes.core`.
- Various bug fixes and documentation improvements.

v0.8.0 (2020-02-12)

- First release of the hybrid symbolic-numeric framework in ANDES.
- A new framework is used to describe DAE models, generate equation documentation, and generate code for numerical simulation.
- Models are written in the new framework. Supported models include GENCLS, GENROU, EXDC2, TGOV1, TG2
- PSS/E raw parser, MATPOWER parser, and ANDES xlsx parser.
- Newton-Raphson power flow, trapezoidal rule for numerical integration, and full eigenvalue analysis.

v0.6.9 (2020-02-12)

- Version 0.6.9 is the last version for the numeric-only modeling framework.
- This version will not be updated any more. But, models, routines and functions will be ported to the new version.

CHAPTER

SIX

VERIFICATION

This section presents the verification of the models and algorithms implemented in ANDES by comparing time-domain simulation results with commercial tools.

6.1 Overview

ANDES produces identical results for the IEEE 14-bus and the NPCC systems with several models. For the CURENT WECC system, ANDES, TSAT, and PSS/E produce slightly different results. In fact, results from different tools can hardly match for large systems with a variety of dynamic models.

ANDES models are verified against:

- PSS/E simulation results
- TSAT simulation results
- Published benchmark data

6.2 Test Cases

6.2.1 IEEE 14-Bus System

- Compact test system with synchronous generators and exciters
- Verified against PSS/E and TSAT
- Results show excellent agreement

6.2.2 NPCC System

- Northeast Power Coordinating Council test system
- Multi-area system with various generator models
- Verified against commercial software

6.2.3 WECC System

- Western Electricity Coordinating Council test system
- Large-scale system with renewable generation
- Results show minor differences due to model implementation variations

6.3 Verification Approach

1. **Model Parameters**: Use identical parameters across all tools
2. **Disturbance**: Apply the same fault or disturbance
3. **Time Step**: Use comparable integration time steps
4. **Comparison Variables**: Compare rotor angles, speeds, and bus voltages

6.4 Detailed Results

The following notebooks contain detailed verification results with comparison plots:

6.4.1 IEEE 14-Bus Verification

Prepared by Hantao Cui. Last revised 23 May 2020.

Background

Two line trip scenarios are used to verify ANDES simulation results with DSATools TSAT.

Dynamic data is created to utilize the available models, including GENROU, TGOV1, IEEEG1, EXST2, EXDC2, ESST3A, IEEEST and ST2CUT. Test case data can be found at the end of the notebook.

Simulation Parameters

Integretion method: Trapezidal Rule (ANDES and TSAT).

Time step size: `1/120` sec. (Note: step size between `1/30` to `1/120` has little impact on the ANDES results. One can use `tstep=1/30` to obtain almost the same results.)

Load conversion: static loads are converted to 100% constant impedances for both P and Q.

TSAT automatic parameter correction is disabled.

Initialization

Power flow solutions are identical across all the two software.

GENROU initialization (E_{FD}, E_{TERM}, P, Q, δ, I_d and I_q) is identical to that from **PSS/E** for all cases (with and without generator saturation). Note that I_d and I_q are in machine base in PSS/E but in system base in ANDES.

GENROU initialization (including all the internal variables E'_d, E'_q, ψ_{kd}, ψ_{kq}, ψ''_d, and ψ''_q) is identical to that from **OpenIPSL**.

No controller limit violation occurs during initialization.

Conclusion

IEEE 14-bus system simulation results from ANDES are identical to that from TSAT.

```
import andes
import numpy as np
from andes.utils.tsat import tsat_to_df, plot_comparison, run_cmp

andes.config_logger(stream_level=30)
```

```
# load scenario 1 data
omega_lt2 = tsat_to_df('omega_lt2s.xls')
v_lt2 = tsat_to_df('v_lt2s.xls')

# load scenario 2 data
omega_gt = tsat_to_df('omega_gt.xls')
v_gt = tsat_to_df('v_gt.xls')
```

Scenario 1: Line 1-2 trips at 1 sec. and reconnects after 2.0 sec.

The reconnection delay is set to 2 seconds to trigger a large disturbance to the system to verify nonlinear models.

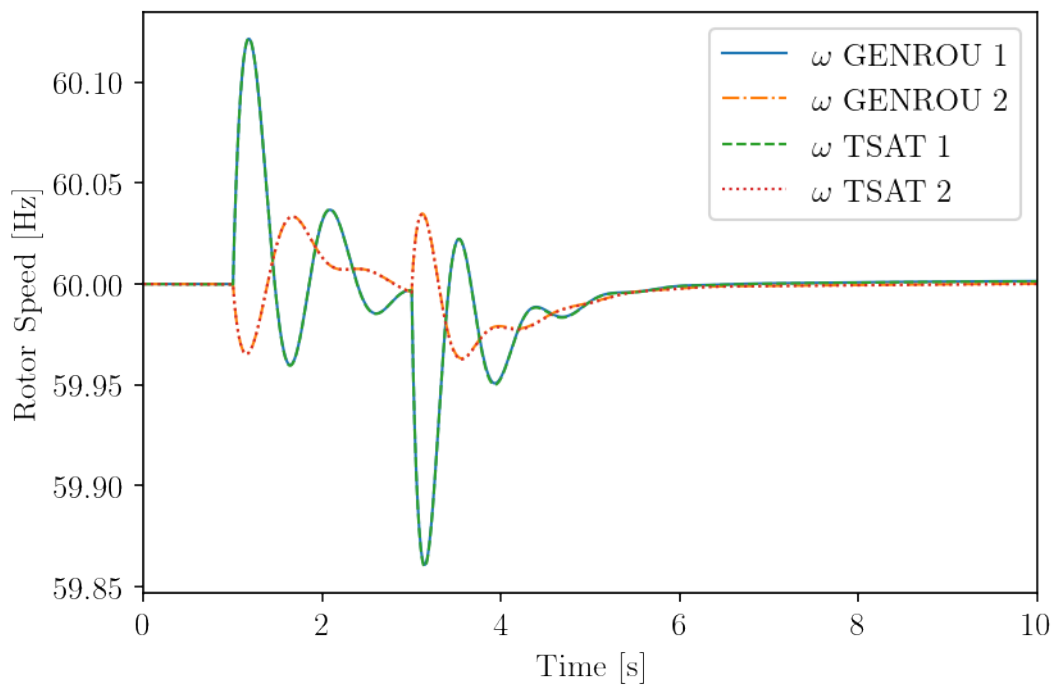
Simulation Setup

```
ss = run_cmp('ieee14.raw', dyr='ieee14.dyr', fault_line='Line_1',
             t1=1.0, t2=3.0, tstep=1/120)
```

```
<Toggle Toggle_1>: Line.Line_1 status changed to 0.0 at t=1.0 sec.
<Toggle Toggle_2>: Line.Line_1 status changed to 1.0 at t=3.0 sec.
100%|██████████████████████████████████| 100/100 [00:16<00:00,  6.15%/s]
```

Scenario 1 Plots

```
fig, ax = plot_comparison(ss, ss.GENROU.omega,
                omega_lt2, a=[0, 1],
                ylabel="Rotor Speed [Hz]",
                tsat_header=[r'$\omega$ TSAT 1', r'$\omega$ TSAT 2'],
                scale=60, left=0, right=10)
```

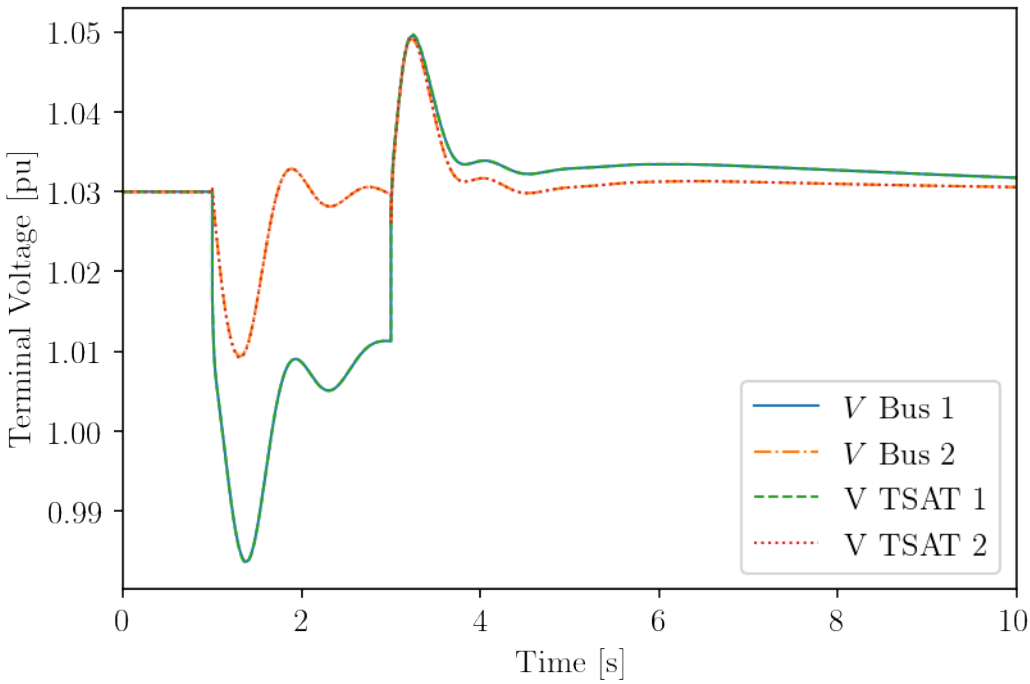



```
fig, ax = plot_comparison(ss, ss.GENROU.v,
                v_lt2, a=[0, 1],
                ylabel="Terminal Voltage [pu]",
```

(continues on next page)

(continued from previous page)

```
                    tsat_header=['V TSAT 1', 'V TSAT 2'],
                    left=0, right=10)
```



Scenario 2: Line 7-8 trips at 1 sec. and reconnects after 0.1 sec.

Line 7-8 is an equivalent branch in of the three-winding transformer. Tripping Line 7-8 will isolate Bus 8.

Simulation Setup

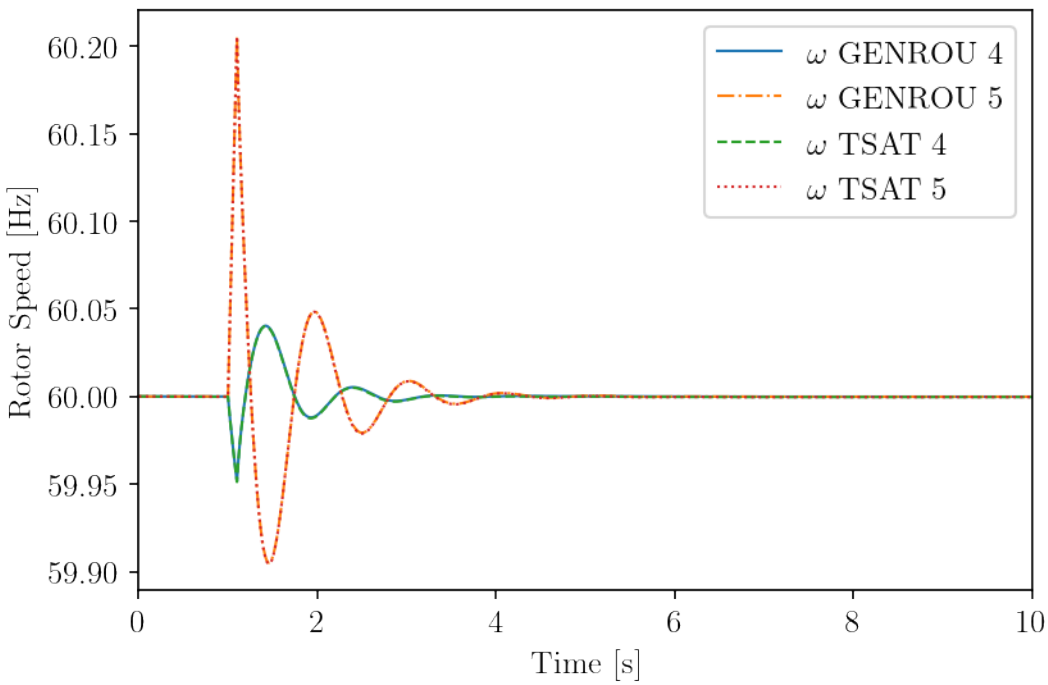
```
ss2 = run_cmp('ieee14.raw', dyr='ieee14.dyr', fault_line='Line_20',
              t1=1.0, t2=1.1, tstep=1/120)
```

```
<Toggle Toggle_1>: Line.Line_20 status changed to 0.0 at t=1.0 sec.
<Toggle Toggle_2>: Line.Line_20 status changed to 1.0 at t=1.1 sec.
100%|██████████████████████████████████| 100/100 [00:12<00:00,  8.10%/s]
```

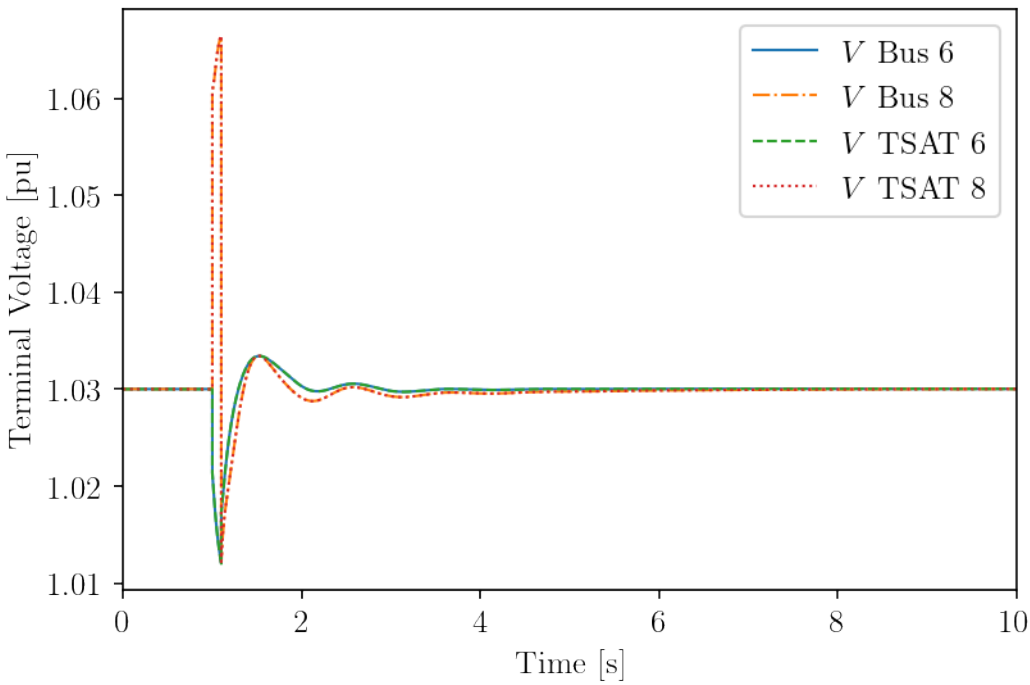
```
# line data
ss2.Line.as_df().iloc[19]
```

Scenario 2 Plots

```
tsat_omega_headers = [ss.TDS.plt._fname[i + 1] for i in ss.GENROU.omega.a]
tsat_omega_headers = [i.replace('GENROU', 'TSAT') for i in tsat_omega_headers]

fig2, ax2 = plot_comparison(ss2, ss2.GENROU.omega,
                omega_gt, a=[3, 4],
                ylabel="Rotor Speed [Hz]",
                tsat_header=tsat_omega_headers,
                scale=60, left=0, right=10)
```




```
tsat_v_headers = [ss.TDS.plt._fname[i + 1 + ss.dae.n] for i in ss.GENROU.v.a]
tsat_v_headers = [i.replace('Bus', 'TSAT') for i in tsat_v_headers]

fig2, ax2 = plot_comparison(ss2, ss2.GENROU.v,
                v_gt, a=[3, 4],
                ylabel="Terminal Voltage [pu]",
                tsat_header=tsat_v_headers,
                left=0, right=10)
```



Appendix: IEEE 14-Bus System Data

Power Flow Data (PSS/E RAW format)

```
# !cat results won't save to PDF

data = !cat ieee14.raw
print('\n'.join(data))
```

```
0,   100.00,  32, 0, 1, 60.00     / PSS(R)E 32 RAW created by rawd32  SUN,␣
↪MAY 17 2020  20:38
 IEEE 14 BUS TEST CASE
DISTRIBUTED WITH ANDES (HTTPS://GITHUB.COM/CURENT/ANDES)
     1,'BUS1        ',  69.0000,3,   1,   1,   1,1.03000,   0.0000
     2,'BUS2        ',  69.0000,2,   1,   1,   1,1.01970,  -1.6032
     3,'BUS3        ',  69.0000,2,   1,   1,   1,1.00042,  -3.4433
     4,'BUS4        ',  69.0000,1,   1,   1,   1,0.99858,  -4.2812
     5,'BUS5        ',  69.0000,1,   1,   1,   1,1.00443,  -3.6850
     6,'BUS6        ', 138.0000,2,   2,   2,   2,0.99871,  -6.3024
     7,'BUS7        ', 138.0000,1,   2,   2,   2,1.00682,  -4.8292
     8,'BUS8        ',  69.0000,2,   2,   2,   2,1.01895,  -1.3945
     9,'BUS9        ', 138.0000,1,   2,   2,   2,1.00193,  -7.3053
    10,'BUS10       ', 138.0000,1,   2,   2,   2,0.99351,  -7.4600
    11,'BUS11       ', 138.0000,1,   2,   2,   2,0.99245,  -7.0444
    12,'BUS12       ', 138.0000,1,   2,   2,   2,0.98639,  -7.3874
    13,'BUS13       ', 138.0000,1,   2,   2,   2,0.98403,  -7.6654
    14,'BUS14       ', 138.0000,1,   2,   2,   2,0.99063,  -9.5636
```

(continues on next page)

(continued from previous page)

```
 0 /End of Bus data, Begin Load data
     2,'1 ',1,   1,   1,    21.700,    12.700,     0.000,     0.000,     0.
↪000,     0.000,   1,1
     3,'1 ',1,   1,   1,    50.000,    25.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
     4,'1 ',1,   1,   1,    47.800,    10.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
     5,'1 ',1,   1,   1,     7.600,     1.600,     0.000,     0.000,     0.
↪000,     0.000,   1,1
     6,'1 ',1,   2,   2,    15.000,     7.500,     0.000,     0.000,     0.
↪000,     0.000,   2,1
     9,'1 ',1,   2,   2,    29.500,    16.600,     0.000,     0.000,     0.
↪000,     0.000,   2,1
    10,'1 ',1,   2,   2,     9.000,     5.800,     0.000,     0.000,     0.
↪000,     0.000,   2,1
    11,'1 ',1,   2,   2,     3.500,     1.800,     0.000,     0.000,     0.
↪000,     0.000,   2,1
    12,'1 ',1,   2,   2,     6.100,     1.600,     0.000,     0.000,     0.
↪000,     0.000,   2,1
    13,'1 ',1,   2,   2,    13.500,     5.800,     0.000,     0.000,     0.
↪000,     0.000,   2,1
    14,'1 ',1,   2,   2,    20.000,     7.000,     0.000,     0.000,     0.
↪000,     0.000,   2,1
 0 /End of Load data, Begin Fixed shunt data
 0 /End of Fixed shunt data, Begin Generator data
     1,'1 ',    81.442,     1.962,   100.000,   -50.000,1.03000,     0,   100.
↪000, 0.00000E+0, 2.30000E-1, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0,  ␣
↪200.000,    50.000,   1,1.0000
     2,'1 ',    40.000,    15.000,    15.000,   -40.000,1.03000,     0,   100.
↪000, 0.00000E+0, 1.30000E-1, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0,   ␣
↪50.000,    10.000,   1,1.0000
     3,'1 ',    40.000,    15.000,    15.000,   -10.000,1.01000,     0,   100.
↪000, 0.00000E+0, 1.30000E-1, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0,   ␣
↪50.000,    10.000,   1,1.0000
     6,'1 ',    30.000,    10.000,    10.000,    -6.000,1.03000,     0,   100.
↪000, 0.00000E+0, 1.20000E-1, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0,   ␣
↪50.000,    10.000,   1,1.0000
     8,'1 ',    35.000,    10.000,    10.000,    -6.000,1.03000,     0,   100.
↪000, 0.00000E+0, 1.20000E-1, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0,   ␣
↪50.000,    10.000,   1,1.0000
 0 /End of Generator data, Begin Branch data
     1,      2,'1 ', 1.93800E-2, 5.91700E-2,   0.05280,  100.00,  100.00,   ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
     1,      5,'1 ', 5.40300E-2, 2.23040E-1,   0.04920,  100.00,  100.00,   ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
     2,      3,'1 ', 4.69900E-2, 1.97970E-1,   0.04380,  100.00,  100.00,   ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
     2,      4,'1 ', 5.81100E-2, 1.76320E-1,   0.03400,  100.00,  100.00,   ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
     2,      5,'1 ', 5.69500E-2, 1.73880E-1,   0.03460,  100.00,  100.00,   ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
     3,      4,'1 ', 6.70100E-2, 1.71030E-1,   0.01280,  100.00,  100.00,   ␣
```

(continues on next page)

(continued from previous page)

```
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
     4,      5,'1 ', 1.33500E-2, 4.21100E-2,   0.00000,  100.00,  100.00,   ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
     6,     11,'1 ', 9.49800E-2, 1.98900E-1,   0.00000,  100.00,  100.00,   ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
     6,     12,'1 ', 1.22910E-1, 2.55810E-1,   0.00000,  100.00,  100.00,   ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
     6,     13,'1 ', 6.61500E-2, 1.30270E-1,   0.00000,  100.00,  100.00,   ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
     7,      9,'1 ',-0.00000E+0, 1.10010E-1,   0.00000,  100.00,  100.00,   ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
     9,     10,'1 ', 3.18100E-2, 8.45000E-2,   0.00000,  100.00,  100.00,   ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
     9,     14,'1 ', 1.27110E-1, 2.70380E-1,   0.00000,  100.00,  100.00,   ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    10,     11,'1 ', 8.20500E-2, 1.92070E-1,   0.00000,  100.00,  100.00,   ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    12,     13,'1 ', 2.20920E-1, 1.99880E-1,   0.00000,  100.00,  100.00,   ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    13,     14,'1 ', 1.70930E-1, 3.48020E-1,   0.00000,  100.00,  100.00,   ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
 0 /End of Branch data, Begin Transformer data
     4,      7,      0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'                 ␣
↪                     ',1,   1,1.0000
 0.00000E+0, 2.09120E-1,   100.00
0.99677,   0.000,   0.000,    20.00,    20.00,     0.00,-1,      0, 1.10000,␣
↪0.90000, 1.10000, 0.90000,  32, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
     4,      9,      0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'                 ␣
↪                     ',1,   1,1.0000
 0.00000E+0, 5.56180E-1,   100.00
0.99677,   0.000,   0.000,    20.00,    20.00,     0.00,-1,      0, 1.10000,␣
↪0.90000, 1.10000, 0.90000,  32, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
     6,      5,      0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,1,'                 ␣
↪                     ',1,   1,1.0000
 0.00000E+0, 2.52020E-1,   100.00
0.99677,   0.000,   0.000,    50.00,    50.00,     0.00,-1,      0, 1.10000,␣
↪0.90000, 1.10000, 0.90000,  32, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
     8,      7,      0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,1,'                 ␣
↪                     ',1,   1,1.0000
 0.00000E+0, 1.76150E-1,   100.00
0.99677,   0.000,   0.000,    50.00,    50.00,     0.00,-1,      0, 1.10000,␣
↪0.90000, 1.10000, 0.90000,  32, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
 0 /End of Transformer data, Begin Area interchange data
   1,      2,      0.000,    999.990,'AREA1       '
   2,      0,      0.000,     10.000,'AREA2       '
 0 /End of Area interchange data, Begin Two-terminal dc line data
 0 /End of Two-terminal dc line data, Begin VSC dc line data
 0 /End of VSC dc line data, Begin Impedance correction table data
```

(continues on next page)

(continued from previous page)

```
 0 /End of Impedance correction table data, Begin Multi-terminal dc line data
 0 /End of Multi-terminal dc line data, Begin Multi-section line data
 0 /End of Multi-section line data, Begin Zone data
   1,'ZONE1       '
   2,'ZONE2       '
 0 /End of Zone data, Begin Inter-area transfer data
 0 /End of Inter-area transfer data, Begin Owner data
   1,'OWNER1      '
   2,'OWNER2      '
 0 /End of Owner data, Begin FACTS device data
 0 /End of FACTS device data, Begin Switched shunt data
     9,1,0,1,1.02500,0.96000,     0,  100.0,'            ',   19.00, 3,   5.
↪00, 1,   4.00
    14,1,0,1,1.02500,0.96000,     0,  100.0,'            ',   15.00, 3,   5.00
 0 /End of Switched shunt data, Begin GNE device data
 0 /End of GNE device data
Q
```

Dynamic Data (PSS/E DYR format)

```
data = !cat ieee14.dyr
print('\n'.join(data))
```

```
      1 'GENROU' 1     6.5000      0.60000E-01  0.20000      0.50000E-01
          4.0000       0.0000       1.8000       1.7500      0.60000
         0.80000      0.23000      0.15000      0.90000E-01  0.38000      /
      1 'ST2CUT' 1          1            0            0            0
          0.0000       0.0000       0.0000       0.0000       30.000
          30.000      0.23000      0.25000E-01  0.23000      0.25000E-01
          0.0000       0.0000      0.60000E-01 -0.60000E-01   0.0000
          0.0000      /
      1 'ESST3A' 1    0.20000E-01  0.20000     -0.20000       8.0000
          1.0000       5.0000       20.000       0.0000       99.000
         -99.000       1.0000       3.6700      0.43500       5.4800
         0.10000E-01  0.98000E-02   3.8600       3.3300      0.40000
          99.000       0.0000    /
      1 'TGOV1'  1    0.50000E-01  0.50000E-01   1.0500      0.30000
          1.0000       2.1000       0.0000      /
      2 'GENROU' 1     6.5000      0.60000E-01  0.20000      0.50000E-01
          6.5000       0.0000       1.8000       1.7500      0.60000
         0.80000      0.28000      0.15000      0.90000E-01  0.38000      /
      2 'ST2CUT' 1          1            0            0            0
          10.000       0.0000       0.0000       0.0000       3.0000
          3.0000      0.15000      0.50000E-01  0.15000      0.50000E-01
         0.15000      0.50000E-01  0.50000E-01 -0.50000E-01   0.0000
          0.0000      /
      2 'EXST1'  1    0.20000E-01   99.000      -99.000       0.0000
         0.20000E-01   50.000      0.20000E-01   9999.0      -9999.0
          0.0000      0.10000E-01   1.0000    /
```

(continues on next page)

(continued from previous page)

```
     2 'IEEEG1' 1          0           0        20.000      0.10000
         0.0000      0.20000       1.0000      -1.0000      0.95000
         0.0000      0.10000       0.0000       0.0000       0.0000
         0.0000       0.0000       0.0000      0.30000       0.0000
         8.7200      0.70000       0.0000    /
     3 'GENROU' 1     6.5000      0.60000E-01  0.20000      0.50000E-01
         5.0000       0.0000       1.8000       1.7500      0.60000
        0.80000      0.34000      0.15000      0.90000E-01  0.38000      /
     3 'IEEEST' 1          3            0       0.0000       0.0000
         0.0000       0.0000       0.0000       0.0000       0.0000
         0.0000       0.0000      0.75000       1.0000       4.2000
        -2.0000      0.10000     -0.10000       0.0000       0.0000      /
     3 'ESST3A' 1    0.20000E-01  0.20000     -0.20000       8.0000
         1.0000       5.0000       20.000       0.0000       99.000
        -99.000       1.0000       3.6700      0.43500       5.4800
        0.10000E-01  0.98000E-02   3.8600       3.3300      0.40000
         99.000       0.0000    /
     3 'IEEEG1' 1          0           0        20.000      0.10000
         0.0000      0.20000       1.0000      -1.0000      0.95000
         0.0000      0.10000       0.0000       0.0000       0.0000
         0.0000       0.0000       0.0000      0.30000       0.0000
         8.7200      0.70000       0.0000    /
     6 'GENROU' 1     6.5000      0.60000E-01  0.20000      0.50000E-01
         5.0000       0.0000       1.8000       1.7500      0.60000
        0.80000      0.28000      0.15000      0.90000E-01  0.38000      /
     6 'ESST3A' 1    0.20000E-01  0.20000     -0.20000       8.0000
         1.0000       5.0000       20.000       0.0000       99.000
        -99.000       1.0000       3.6700      0.43500       5.4800
        0.10000E-01  0.98000E-02   3.8600       3.3300      0.40000
         99.000       0.0000    /
     6 'TGOV1'  1    0.50000E-01  0.50000E-01   1.0500      0.30000
         1.0000       2.1000       0.0000      /
     8 'GENROU' 1     6.5000      0.60000E-01  0.20000      0.50000E-01
         5.0000       0.0000       1.8000       1.7500      0.60000
        0.80000      0.34000      0.15000      0.90000E-01  0.38000      /
     8 'ESST3A' 1    0.20000E-01  0.20000     -0.20000       8.0000
         1.0000       5.0000       20.000       0.0000       99.000
        -99.000       1.0000       3.6700      0.43500       5.4800
        0.10000E-01  0.98000E-02   3.8600       3.3300      0.40000
         99.000       0.0000    /
     8 'TGOV1'  1    0.50000E-01  0.50000E-01   1.0500      0.30000
         1.0000       2.1000       0.0000      /
```

6.4.2 CURENT NPCC Verification

Prepared by Hantao Cui. Last revised 23 May 2020.

Background

The CURENT NPCC test system contains 140 buses, 233 branches, and 48 generators. Dynamic data uses models GENROU, GENCLS, TGOV1 and IEEEX1.

One line trip scenario is used to verify ANDES simulation results with DSATools TSAT.

Simulation Parameters

Integretion method: Trapezidal Rule (ANDES and TSAT).

Time step size: `1/120` sec. (Note: step size between `1/30` to `1/120` has little impact on the ANDES results. One can use `tstep=1/30` to obtain almost the same results.)

Load conversion: static loads are converted to 100% constant impedances for both P and Q.

TSAT automatic parameter correction is disabled.

Initialization

Power flow solutions are identical across all the two software.

GENROU initialization (E_{FD}, E_{TERM}, P, Q, δ, I_d and I_q) is identical to that from **PSS/E** for all cases (with and without generator saturation). Note that I_d and I_q are in machine base in PSS/E but in system base in ANDES.

GENROU initialization (including all the internal variables E'_d, E'_q, ψ_{kd}, ψ_{kq}, ψ''_d, and ψ''_q) is identical to that from **OpenIPSL**.

No controller limit violation occurs during initialization.

Conclusion

NPCC simulation results from ANDES are identical to that from TSAT.

```
import andes
import numpy as np
from andes.utils.tsat import tsat_to_df, plot_comparison, run_cmp

andes.config_logger(stream_level=30)
```

```
omega_lt2s = tsat_to_df('omega_lt2s.xls')
v_lt2s = tsat_to_df('v_lt2s.xls')
```

Scenario 1: Line 1-2 trips at 1 sec. and reconnects after 2.0 sec.

The reconnection delay is set to 2 seconds to trigger a large disturbance to the system to verify nonlinear models.

Simulation Setup

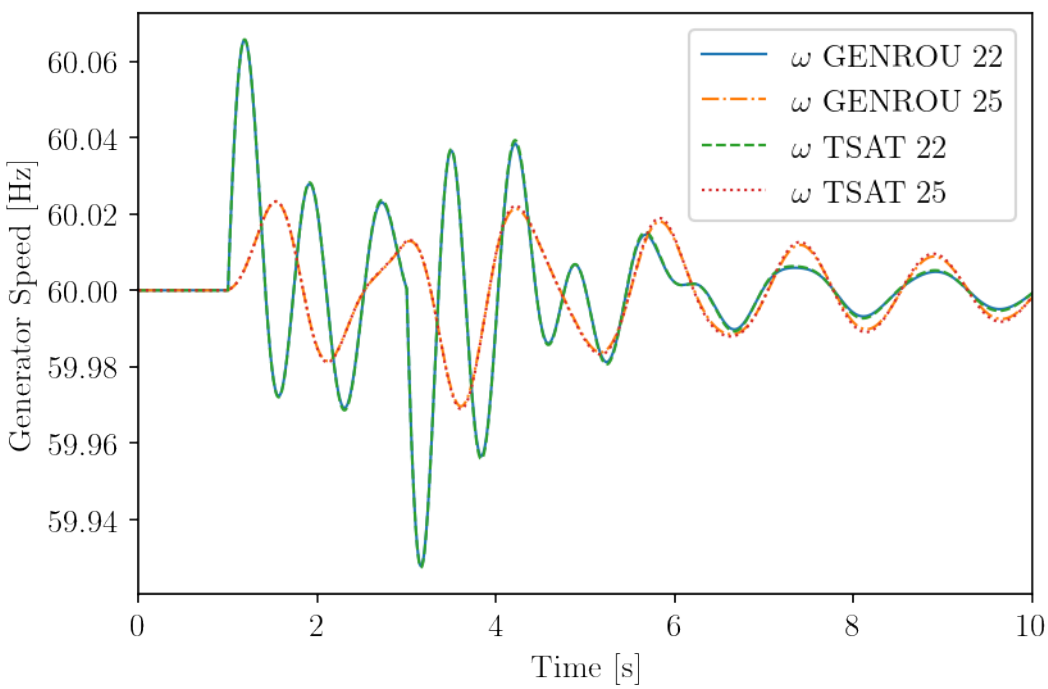
```
ss = run_cmp('npcc.raw', dyr='npcc.dyr', fault_line='Line_1',
             t1=1.0, t2=3.0, tstep=1/120)
```

```
<Toggle Toggle_1>: Line.Line_1 status changed to 0.0 at t=1.0 sec.
<Toggle Toggle_2>: Line.Line_1 status changed to 1.0 at t=3.0 sec.
100%|██████████████████████████████████| 100/100 [00:10<00:00,  9.87%/s]
```

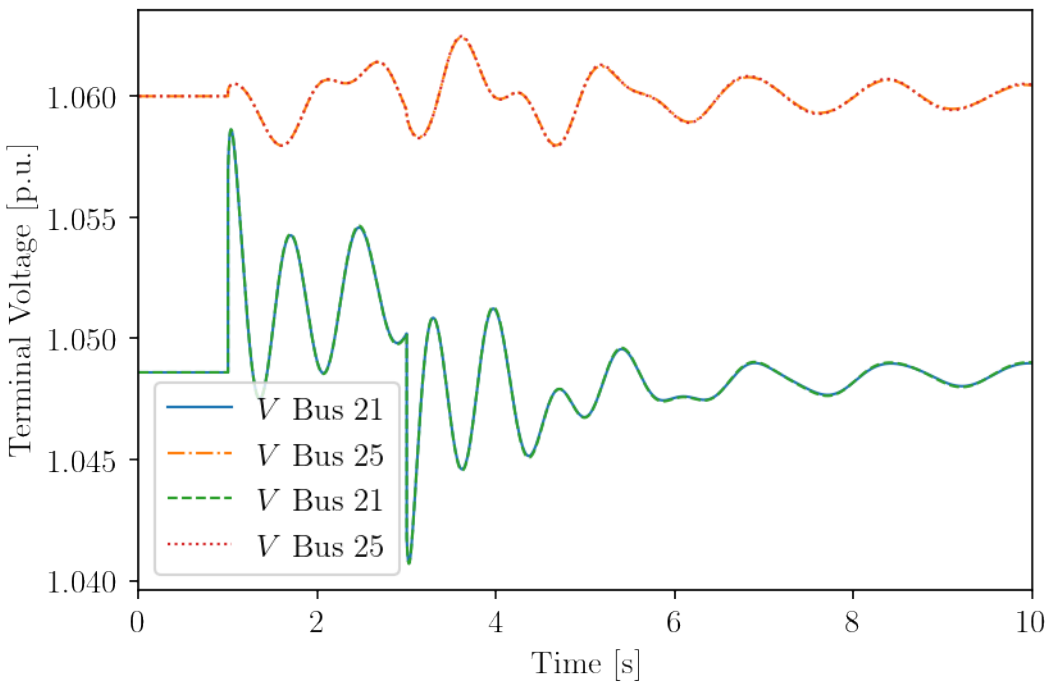
```
# line information
ss.Line.as_df(vin=True).iloc[0]
```

Scenario 1 Plots

```
tsat_omega_headers = [ss.TDS.plt._fname[i + 1] for i in ss.GENROU.omega.a]
tsat_omega_headers = [i.replace('GENROU', 'TSAT') for i in tsat_omega_headers]

fig, ax = plot_comparison(ss, ss.GENROU.omega, omega_lt2s,
                a=[0, 3], ylabel="Generator Speed [Hz]",
                tsat_header=tsat_omega_headers,
                scale=60, right=10)
```



```
tsat_v_headers = [ss.TDS.plt._fname[i + 1 + ss.dae.n] for i in ss.GENROU.v.a]
tsat_v_headers = [i.replace('GENROU', 'TSAT') for i in tsat_v_headers]

fig, ax = plot_comparison(ss, ss.GENROU.v,
                          v_lt2s, a=[0, 5],
                          ylabel="Terminal Voltage [p.u.]",
                          tsat_header=tsat_v_headers, right=10)
```



Appendix: CURENT NPCC 140-Bus System Data

Power Flow Data (PSS/E RAW format)

```
data = !cat npcc.raw
print('\n'.join(data))
```

```
0,   100.00,  32, 0, 1, 60.00     / PSS(R)E 32 RAW created by rawd32  MON,␣
↪JUN 22 2015   8:32
CURENT NPCC 48-MACHINE TESTBED, CONVERTED FROM PST BY BIN WANG
REVISED BY YIN LEI, 11/18/2013
     1,'MILLSTONE PT', 345.0000,1,   1,   1,   1,1.01517,   4.8434
     2,'MONTVILLE   ', 345.0000,1,   1,   1,   1,1.01090,   4.0566
     3,'MONTVILLE   ', 115.0000,1,   1,   1,   1,0.99170,   4.0069
     4,'MONTVILLE   ', 345.0000,1,   1,   1,   1,1.01125,   4.1005
     5,'CARD ST     ', 345.0000,1,   1,   1,   1,1.00618,   2.3527
     6,'SHERMAN RD  ', 345.0000,1,   1,   1,   1,1.00594,   1.8603
     7,'MEDWAY      ', 345.0000,1,   1,   1,   1,1.02071,   3.2654
     8,'MILLBURY    ', 115.0000,1,   1,   1,   1,1.02151,   1.9975
```

(continues on next page)

(continued from previous page)

```
 9,'CARPENTER HL', 115.0000,1,   1,   1,   1,1.01811,   1.0727
10,'BRAYTON PT  ', 115.0000,1,   1,   1,   1,1.03060,   8.0176
11,'BRAYTON PT  ', 115.0000,1,   1,   1,   1,1.01169,   6.7075
12,'BRIDGEWATER ', 115.0000,1,   1,   1,   1,1.02371,   5.9431
13,'PILGRIM     ', 345.0000,1,   1,   1,   1,1.04565,  10.6737
14,'CANAL       ', 345.0000,1,   1,   1,   1,1.03950,  10.3866
15,'WALPOLA     ', 345.0000,1,   1,   1,   1,1.02697,   3.4395
16,'VERMONT YNK ', 345.0000,1,   1,   1,   1,1.04108,   4.4713
17,'SCOBIE      ', 345.0000,1,   1,   1,   1,1.04016,   3.2759
18,'SANDY POND  ', 115.0000,1,   1,   1,   1,1.02570,   1.5065
19,'POWNAL      ', 345.0000,1,   1,   1,   1,1.04132,   6.4689
20,'MAINE YANKEE', 345.0000,1,   1,   1,   1,1.04211,   9.1477
21,'MILLSTONE PT',  24.0000,2,   1,   1,   1,1.04860,  11.8582
22,'BRAYTON PT  ',  18.0000,2,   1,   1,   1,1.05930,  12.6562
23,'BRAYTON PT  ',  20.0000,2,   1,   1,   1,1.01570,  11.7526
24,'PILGRIM     ',  22.8000,2,   1,   1,   1,1.07625,  15.7768
25,'CANAL       ',  20.0000,2,   1,   1,   1,1.06000,  18.3042
26,'MAINE YANKEE',  20.0000,2,   1,   1,   1,1.04550,  15.7134
27,'VERMONT YNK ',  22.0000,2,   1,   1,   1,1.04550,  11.0690
28,'NORTHFIELD  ',  13.8000,1,   1,   1,   1,1.02751,   3.2234
29,'NORTHFIELD  ', 345.0000,1,   1,   1,   1,1.02751,   3.2234
30,'LUDLOW      ', 345.0000,1,   1,   1,   1,1.01622,   0.7209
31,'MANCHESTER  ', 345.0000,1,   1,   1,   1,0.99761,   0.3966
32,'SCOVILL ROCK', 345.0000,1,   1,   1,   1,1.00329,   1.8450
33,'HADDAM NECK ', 345.0000,1,   1,   1,   1,1.00627,   2.5437
34,'BLACK POND  ', 345.0000,1,   1,   1,   1,0.99603,   0.4986
35,'SOUTHINGTEN ', 345.0000,1,   1,   1,   1,0.99524,   0.0743
36,'CONN YANKEE ', 345.0000,2,   1,   1,   1,1.04860,   9.9630
37,'NEW SEATHED ', 345.0000,1,   2,   1,   1,1.04419,   4.6062
38,'GILBOA      ', 345.0000,1,   2,   1,   1,1.04210,   6.9752
39,'LEEDS       ', 345.0000,1,   2,   1,   1,1.04100,   3.2739
40,'NEW SEATHED ', 230.0000,1,   2,   1,   1,1.03072,   0.5762
41,'ROTTERDAM   ', 115.0000,1,   2,   1,   1,1.02634,  -3.3709
42,'ALBANY      ', 115.0000,2,   2,   1,   1,1.04000,  -2.5665
43,'EDIC        ', 345.0000,1,   2,   1,   1,1.04908,   8.3430
44,'PORTER      ', 230.0000,1,   2,   1,   1,1.04792,   6.5696
45,'PORTER      ', 115.0000,1,   2,   1,   1,1.04349,   3.9472
46,'COLTON      ', 115.0000,1,   2,   1,   1,1.04781,   6.1850
47,'MOSES W     ', 230.0000,2,   2,   1,   1,1.03000,  11.6793
48,'MOSES E     ', 230.0000,2,   2,   1,   1,1.05000,  15.0127
49,'PLATTSBURGH ', 115.0000,1,   2,   1,   1,0.96523,  -1.1162
50,'CLAY        ', 345.0000,2,   2,   1,   1,1.05000,  13.2144
51,'CLAY        ', 115.0000,2,   2,   1,   1,1.04000,   8.5283
52,'ROCHESTER   ', 345.0000,1,   2,   1,   1,1.04193,  19.4130
53,'ROCHESTER   ', 115.0000,2,   2,   1,   1,1.03000,  18.0053
54,'NIAGARA W   ', 230.0000,2,   2,   1,   1,1.04080,  26.9321
55,'NIAGARA E   ', 230.0000,2,   2,   1,   1,1.04000,  26.2095
56,'HUNTLEY     ', 230.0000,2,   2,   1,   1,1.03000,  26.8995
57,'HUNTLEY     ', 115.0000,2,   2,   1,   1,1.03000,  25.7233
58,'GARDENVILLE ', 230.0000,1,   2,   1,   1,1.01010,  24.5883
59,'GARDENVILLE ', 115.0000,1,   2,   1,   1,0.99669,  22.1857
60,'DUNKIRK     ', 230.0000,2,   2,   1,   1,1.04000,  30.2181
```

(continues on next page)

(continued from previous page)

```
  61,'DUNKIRK     ', 115.0000,2,   2,   1,   1,1.04000,  26.7270
  62,'STOLLE RD   ', 230.0000,1,   2,   1,   1,1.01628,  23.9633
  63,'MEYER       ', 230.0000,1,   2,   1,   1,1.00548,  17.6467
  64,'MEYER       ', 115.0000,1,   2,   1,   1,1.00715,  14.5476
  65,'GESONIDGE   ', 115.0000,2,   2,   1,   1,1.04000,  15.7558
  66,'WATERCURE   ', 345.0000,1,   2,   1,   1,1.01975,  20.8736
  67,'WRHL        ', 230.0000,1,   2,   1,   1,1.00265,  16.6813
  68,'HILLSIDE    ', 115.0000,2,   2,   1,   1,0.99000,  12.7432
  69,'BINGHAMTON  ', 345.0000,1,   2,   1,   1,1.01948,  14.3356
  70,'BINGHAMTON  ', 230.0000,1,   2,   1,   1,0.99894,  13.5662
  71,'BINGHAMTON  ', 115.0000,2,   2,   1,   1,1.00000,  11.1142
  72,'LAPEER      ', 115.0000,2,   2,   1,   1,1.05000,  10.1183
  73,'PLEASANT VLY', 345.0000,1,   3,   1,   1,1.03292,   1.6376
  74,'MILLWOOD    ', 345.0000,1,   3,   1,   1,1.02708,   0.2979
  75,'RAMAPO      ', 138.0000,1,   3,   1,   1,1.04944,   1.2631
  76,'RAMAPO      ', 345.0000,1,   3,   1,   1,1.04639,   0.7974
  77,'BUCHANAN    ', 345.0000,1,   3,   1,   1,1.03539,   0.5458
  78,'CE UG       ', 345.0000,3,   3,   1,   1,1.02000,   0.0000
  79,'RAV A-3     ',  22.0000,2,   3,   1,   1,1.05000,   8.1715
  80,'NORTHPORT   ', 138.0000,2,   3,   1,   1,1.02320,   0.6309
  81,'GOETHALS    ', 345.0000,1,   3,   1,   1,1.02198,  -3.2846
  82,'AK-3        ',  22.0000,2,   3,   1,   1,1.05000,   7.5680
  83,'KEITH       ', 220.0000,1,   4,   1,   1,0.95576,  22.9842
  84,'KEITH JCT   ', 118.1000,1,   4,   1,   1,0.96323,  23.4874
  85,'LAMBTON     ', 220.0000,1,   4,   1,   1,0.99128,  36.4486
  86,'LAMBTON     ',  24.0000,2,   4,   1,   1,1.00000,  41.7274
  87,'LAMBTON     ', 220.0000,1,   4,   1,   1,1.01813,  33.6520
  88,'SCOTT       ', 220.0000,1,   4,   1,   1,0.97335,  32.8563
  89,'SCOTT       ', 220.0000,1,   4,   1,   1,0.95584,  30.2514
  90,'VIDA        ', 220.0000,1,   4,   1,   1,0.95411,  30.1020
  91,'RICHVIEW    ', 220.0000,2,   4,   1,   1,1.04000,  11.2134
  92,'HANMER      ', 500.0000,2,   4,   1,   1,1.04000,  18.6819
  93,'ESSA        ', 220.0000,1,   4,   1,   1,0.96211,   7.5454
  94,'ORANGEVILLE ', 220.0000,1,   4,   1,   1,0.95948,   7.7792
  95,'DES JOAC    ', 220.0000,1,   4,   1,   1,0.97264,  14.6338
  96,'HOLDEN      ',  13.8000,1,   4,   1,   1,1.02323,  15.2006
  97,'MARTINDELE  ', 220.0000,2,   4,   1,   1,1.04000,  14.0482
  98,'STLA        ', 220.0000,2,   4,   1,   1,1.04000,  31.9721
  99,'PVR         ', 220.0000,1,   4,   1,   1,1.03842,  31.3229
 100,'PHAS        ', 220.0000,1,   4,   1,   1,1.03248,  26.3181
 101,'SAB 2       ', 220.0000,2,   4,   1,   1,1.05000,  24.3484
 102,'BP 76       ', 220.0000,1,   4,   1,   1,1.04550,  25.5246
 103,'PA 27       ', 220.0000,1,   4,   1,   1,1.04629,  25.3162
 104,'BEACH       ', 220.0000,1,   4,   1,   1,0.99699,  13.1740
 105,'BUCHANAN    ', 220.0000,1,   4,   1,   1,0.97457,  24.1225
 106,'NEALE       ', 220.0000,1,   4,   1,   1,0.99289,  14.5859
 107,'NEALE       ', 220.0000,1,   4,   1,   1,1.00616,  15.9332
 108,'S CT        ', 220.0000,1,   4,   1,   1,1.00033,  13.9155
 109,'SLT         ', 220.0000,1,   4,   1,   1,1.00029,  13.8910
 110,'BURLINGTON  ', 220.0000,1,   4,   1,   1,1.00281,  12.4399
 111,'DETWIERER   ', 220.0000,1,   4,   1,   1,0.96194,  10.0383
 112,'CHATHAM     ', 220.0000,1,   4,   1,   1,0.97541,  29.6056
```

(continues on next page)

(continued from previous page)

```
  113,'LAUZON      ', 118.1000,1,   4,   1,   1,0.95230,  22.4473
  114,'MARYSVILLE  ', 220.0000,1,   5,   1,   1,0.95972,  30.6392
  115,'ST CLAIR    ', 500.0000,2,   5,   1,   1,1.02000,  33.4478
  116,'WATERMAN    ', 220.0000,1,   5,   1,   1,0.99338,  26.8500
  117,'PONTIAC     ', 230.0000,1,   5,   1,   1,1.00506,  30.8651
  118,'WAYNE       ', 115.0000,1,   5,   1,   1,0.99857,  31.7891
  119,'THETFORD    ', 220.0000,2,   5,   1,   1,1.00000,  30.2408
  120,'ARGENTA     ', 220.0000,2,   5,   1,   1,1.02000,  26.6696
  121,'SC415       ', 220.0000,2,   5,   1,   1,1.04000,  32.8926
  122,'WATERMAN    ', 220.0000,2,   5,   1,   1,1.05000,  32.3212
  123,'MONROE      ', 230.0000,2,   5,   1,   1,1.01000,  37.6123
  124,'BRANCHBURG  ', 230.0000,1,   6,   1,   1,1.04471,   2.3841
  125,'LINDON      ', 230.0000,1,   6,   1,   1,1.02196,  -3.4160
  126,'HUDSON      ', 230.0000,1,   6,   1,   1,1.02261,  -5.3945
  127,'ALBURTIS    ', 500.0000,1,   6,   1,   1,1.04853,   6.5520
  128,'WHITPAIN    ', 500.0000,1,   6,   1,   1,1.03926,   2.7377
  129,'KEENEY      ', 500.0000,1,   6,   1,   1,1.02303,   8.9401
  130,'PEACH BOTTOM', 500.0000,2,   6,   1,   1,1.02000,  12.1966
  131,'CONASTONE   ', 500.0000,1,   6,   1,   1,1.02604,  13.4125
  132,'JUNIATA     ', 500.0000,1,   6,   1,   1,1.03978,  19.5565
  133,'CONEMAUGH   ', 500.0000,2,   6,   1,   1,1.02000,  34.3480
  134,'HOMER CITY  ', 230.0000,2,   6,   1,   1,1.03000,  25.5764
  135,'KEYSTONE    ', 500.0000,2,   6,   1,   1,1.02000,  35.0063
  136,'ERIE W      ', 345.0000,1,   6,   1,   1,1.00058,  19.7176
  137,'ERIE S      ', 230.0000,2,   6,   1,   1,1.04000,  20.3245
  138,'E TOMANO    ', 230.0000,1,   6,   1,   1,0.97073,  13.5527
  139,'KYGER       ', 345.0000,2,   6,   1,   1,1.01000,  29.3780
  140,'TE          ', 230.0000,1,   6,   1,   1,1.04132,  30.2102
0 /End of Bus data, Begin Load data
    3,'1 ',1,   1,   1,     9.000,    88.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
    6,'1 ',1,   1,   1,   320.000,   153.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
    7,'1 ',1,   1,   1,   329.000,    32.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
    9,'1 ',1,   1,   1,   158.000,    30.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
   11,'1 ',1,   1,   1,   680.000,   103.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
   12,'1 ',1,   1,   1,   274.000,   115.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
   14,'1 ',1,   1,   1,   248.000,    85.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
   15,'1 ',1,   1,   1,   309.000,   -92.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
   16,'1 ',1,   1,   1,   224.000,    47.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
   17,'1 ',1,   1,   1,   139.000,    17.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
   18,'1 ',1,   1,   1,   281.000,    76.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
   19,'1 ',1,   1,   1,   206.000,    28.000,     0.000,     0.000,     0.
```

(continues on next page)

(continued from previous page)

```
↪000,     0.000,   1,1
    20,'1 ',1,   1,   1,   284.000,    27.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
    30,'1 ',1,   1,   1,   322.000,     2.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
    31,'1 ',1,   1,   1,   500.000,   184.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
    34,'1 ',1,   1,   1,   234.000,    84.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
    35,'1 ',1,   1,   1,   522.000,   177.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
    36,'1 ',1,   1,   1,     9.000,     5.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
    41,'1 ',1,   2,   1,   700.000,   125.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
    42,'1 ',1,   2,   1,   177.000,    17.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
    45,'1 ',1,   2,   1,   256.000,    17.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
    46,'1 ',1,   2,   1,   177.000,    25.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
    46,'2 ',1,   2,   1,  -154.000,   -67.500,     0.000,     0.000,     0.
↪000,     0.000,   1,1
    47,'1 ',1,   2,   1,   528.000,   100.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
    49,'1 ',1,   2,   1,   208.000,    17.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
    51,'1 ',1,   2,   1,   731.000,    75.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
    53,'1 ',1,   2,   1,   923.000,   250.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
    54,'1 ',1,   2,   1,    20.000,     0.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
    55,'1 ',1,   2,   1,   939.000,   132.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
    56,'1 ',1,   2,   1,   172.000,    44.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
    57,'1 ',1,   2,   1,   240.000,    -6.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
    58,'1 ',1,   2,   1,    84.000,    38.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
    59,'1 ',1,   2,   1,   705.000,   198.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
    61,'1 ',1,   2,   1,   237.000,     8.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
    64,'1 ',1,   2,   1,   143.000,    25.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
    65,'1 ',1,   2,   1,    91.000,     0.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
    68,'1 ',1,   2,   1,   211.000,    41.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
    71,'1 ',1,   2,   1,   371.000,    53.000,     0.000,     0.000,     0.
```

(continues on next page)

(continued from previous page)

```
↪000,     0.000,   1,1
    74,'1 ',1,   3,   1,   102.000,     0.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
    78,'1 ',1,   3,   1,  2000.000,   600.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
    80,'1 ',1,   3,   1,   700.000,    25.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
    83,'1 ',1,   4,   1,    80.000,    16.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
    85,'1 ',1,   4,   1,    18.000,     0.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
    86,'1 ',1,   4,   1,    50.000,    25.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
    88,'1 ',1,   4,   1,   110.000,    55.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
    89,'1 ',1,   4,   1,   100.000,    70.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
    90,'1 ',1,   4,   1,   115.000,    45.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
    91,'1 ',1,   4,   1,  1650.000,   500.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
    92,'1 ',1,   4,   1,    10.000,   320.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
    93,'1 ',1,   4,   1,   125.000,     0.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
    94,'1 ',1,   4,   1,    95.000,     0.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
    95,'1 ',1,   4,   1,    25.000,   100.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
    95,'2 ',1,   4,   1,  -125.000,   -13.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
    96,'1 ',1,   4,   1,    20.000,     0.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
    96,'2 ',1,   4,   1,   -60.000,   -25.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
    97,'1 ',1,   4,   1,   910.000,    50.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
   101,'1 ',1,   4,   1,   350.000,   130.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
   104,'1 ',1,   4,   1,   200.000,    90.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
   105,'1 ',1,   4,   1,   325.000,     0.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
   106,'1 ',1,   4,   1,     0.000,   -12.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
   107,'1 ',1,   4,   1,     0.000,   -19.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
   108,'1 ',1,   4,   1,     0.000,   -13.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
   109,'1 ',1,   4,   1,     0.000,   -13.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
   110,'1 ',1,   4,   1,   718.000,   125.000,     0.000,     0.000,     0.
```

(continues on next page)

(continued from previous page)

```
↪000,     0.000,   1,1
  111,'1 ',1,   4,   1,   399.000,    45.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
  113,'1 ',1,   4,   1,   206.000,     0.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
  114,'1 ',1,   5,   1,     0.000,    90.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
  115,'1 ',1,   5,   1,   100.000,     0.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
  116,'1 ',1,   5,   1,   200.000,     0.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
  117,'1 ',1,   5,   1,   400.000,    97.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
  118,'1 ',1,   5,   1,   280.000,    86.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
  119,'1 ',1,   5,   1,   750.000,   200.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
  120,'1 ',1,   5,   1,   946.000,   -40.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
  123,'1 ',1,   5,   1,    63.000,    10.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
  124,'1 ',1,   6,   1,   370.000,    71.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
  124,'2 ',1,   6,   1,     0.000,   -22.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
  125,'1 ',1,   6,   1,   320.000,     5.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
  126,'1 ',1,   6,   1,   450.000,     0.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
  127,'1 ',1,   6,   1,   200.000,     0.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
  127,'2 ',1,   6,   1,     0.000,  -150.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
  128,'1 ',1,   6,   1,   970.000,    60.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
  128,'2 ',1,   6,   1,     0.000,  -162.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
  129,'1 ',1,   6,   1,   610.000,     0.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
  129,'2 ',1,   6,   1,     0.000,   -50.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
  130,'1 ',1,   6,   1,   180.000,     0.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
  131,'1 ',1,   6,   1,  1160.000,   468.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
  131,'2 ',1,   6,   1,     0.000,  -900.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
  132,'1 ',1,   6,   1,   210.000,     0.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
  132,'2 ',1,   6,   1,     0.000,  -337.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
  135,'1 ',1,   6,   1,   170.000,     0.000,     0.000,     0.000,     0.
```

(continues on next page)

(continued from previous page)

```
↪000,     0.000,   1,1
  136,'1 ',1,   6,   1,   270.000,   223.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
  138,'1 ',1,   6,   1,   110.000,    64.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
 0 /End of Load data, Begin Fixed shunt data
 0 /End of Fixed shunt data, Begin Generator data
   21,'1 ',   650.000,   215.117,   999.000,  -999.000,1.04860,     0,   750.
↪000, 0.00000E+0, 2.17500E-1, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000, -9999.000,   1,1.0000
   22,'1 ',   632.000,   208.118,   999.000,  -999.000,1.05930,     0,   700.
↪000, 0.00000E+0, 2.06500E-1, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000, -9999.000,   1,1.0000
   23,'1 ',   276.650,    10.788,   999.000,  -999.000,1.01570,     0,   300.
↪000, 0.00000E+0, 9.99000E-2, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000, -9999.000,   1,1.0000
   23,'2 ',   226.350,     8.827,  9999.000, -9999.000,1.01570,     0,   300.
↪000, 0.00000E+0, 2.23500E-1, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000, -9999.000,   1,1.0000
   24,'1 ',   700.000,   261.533,   999.000,  -999.000,1.07625,     0,   800.
↪000, 0.00000E+0, 2.32000E-1, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000, -9999.000,   1,1.0000
   25,'1 ',   560.000,   108.222,   999.000,  -999.000,1.06000,     0,   650.
↪000, 0.00000E+0, 2.09300E-1, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000, -9999.000,   1,1.0000
   26,'1 ',   800.000,    27.488,   999.000,  -999.000,1.04550,     0,   900.
↪000, 0.00000E+0, 2.81700E-1, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000, -9999.000,   1,1.0000
   27,'1 ',   540.000,    37.044,   999.000,  -999.000,1.04550,     0,   600.
↪000, 0.00000E+0, 1.80000E-1, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000, -9999.000,   1,1.0000
   36,'1 ',   554.000,   217.901,   999.000,  -999.000,1.04860,     0,   650.
↪000, 0.00000E+0, 2.99000E-1, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000, -9999.000,   1,1.0000
   42,'1 ',   370.000,   134.949,   999.000,  -999.000,1.04000,     0,   400.
↪000, 0.00000E+0, 1.78000E-1, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000, -9999.000,   1,1.0000
   47,'1 ',   400.000,    -3.268,   999.000,  -999.000,1.03000,     0,   450.
↪000, 0.00000E+0, 3.51900E-1, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000, -9999.000,   1,1.0000
   48,'1 ',   400.000,   107.452,   999.000,  -999.000,1.05000,     0,   500.
↪000, 0.00000E+0, 3.91000E-1, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000, -9999.000,   1,1.0000
   50,'1 ',   600.000,    41.180,   999.000,  -999.000,1.05000,     0,   700.
↪000, 0.00000E+0, 2.95000E-1, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000, -9999.000,   1,1.0000
   51,'1 ',   141.000,    20.685,   999.000,  -999.000,1.04000,     0,   150.
↪000, 0.00000E+0, 6.19500E-2, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000, -9999.000,   1,1.0000
   53,'1 ',   833.000,   173.977,   999.000,  -999.000,1.03000,     0,   100.
↪000, 0.00000E+0, 2.00000E-2, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000, -9999.000,   1,1.0000
```

(continues on next page)

(continued from previous page)

```
    54,'1 ',   557.500,    -0.649,   999.000,  -999.000,1.04080,     0,   650.
↪000, 0.00000E+0, 3.25000E-1, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000, -9999.000,   1,1.0000
    54,'2 ',   557.500,    -0.649,  9999.000, -9999.000,1.04080,     0,   650.
↪000, 0.00000E+0, 3.25000E-1, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000, -9999.000,   1,1.0000
    55,'1 ',  1095.000,   148.655,   999.000,  -999.000,1.04000,     0,  1200.
↪000, 0.00000E+0, 3.62400E-1, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000, -9999.000,   1,1.0000
    56,'1 ',   410.000,    87.207,   999.000,  -999.000,1.03000,     0,   500.
↪000, 0.00000E+0, 2.57500E-1, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000, -9999.000,   1,1.0000
    57,'1 ',   298.000,     3.534,   999.000,  -999.000,1.03000,     0,   350.
↪000, 0.00000E+0, 1.62400E-1, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000, -9999.000,   1,1.0000
    60,'1 ',   410.000,    54.828,   999.000,  -999.000,1.04000,     0,   450.
↪000, 0.00000E+0, 2.35350E-1, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000, -9999.000,   1,1.0000
    61,'1 ',   194.000,    13.180,   999.000,  -999.000,1.04000,     0,   200.
↪000, 0.00000E+0, 1.89400E-1, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000, -9999.000,   1,1.0000
    65,'1 ',    91.000,    51.625,   999.000,  -999.000,1.04000,     0,   100.
↪000, 0.00000E+0, 2.00000E-2, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000, -9999.000,   1,1.0000
    68,'1 ',    66.000,    -0.140,   999.000,  -999.000,0.99000,     0,   100.
↪000, 0.00000E+0, 2.00000E-2, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000, -9999.000,   1,1.0000
    71,'1 ',   134.000,  -373.660,   999.000,  -999.000,1.00000,     0,   100.
↪000, 0.00000E+0, 2.00000E-2, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000, -9999.000,   1,1.0000
    72,'1 ',   150.000,   379.422,   999.000,  -999.000,1.05000,     0,   100.
↪000, 0.00000E+0, 2.00000E-2, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000, -9999.000,   1,1.0000
    78,'1 ',   466.019,    74.010,   999.000,  -999.000,1.02000,     0,   100.
↪000, 0.00000E+0, 2.00000E-2, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000, -9999.000,   1,1.0000
    79,'1 ',  1000.000,   257.345,   999.000,  -999.000,1.05000,     0,  1150.
↪000, 0.00000E+0, 3.22000E-1, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000, -9999.000,   1,1.0000
    80,'1 ',   720.000,    -0.078,   999.000,  -999.000,1.02320,     0,   800.
↪000, 0.00000E+0, 2.00000E-1, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000, -9999.000,   1,1.0000
    82,'1 ',   460.000,   125.095,   999.000,  -999.000,1.05000,     0,   500.
↪000, 0.00000E+0, 2.50000E-1, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000, -9999.000,   1,1.0000
    86,'1 ',  1650.000,   251.799,   999.000,  -999.000,1.00000,     0,  1900.
↪000, 0.00000E+0, 1.90000E-1, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000, -9999.000,   1,1.0000
    91,'1 ',   930.000,  1082.086,  9999.000, -9999.000,1.04000,     0,   100.
↪000, 0.00000E+0, 2.00000E-2, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000, -9999.000,   1,1.0000
    92,'1 ',   450.000,   324.468,   999.000,  -999.000,1.04000,     0,   100.
```

(continues on next page)

(continued from previous page)

```
↪000, 0.00000E+0, 2.00000E-2, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000, -9999.000,   1,1.0000
    97,'1 ',   600.000,    96.902,   999.000,  -999.000,1.04000,     0,   100.
↪000, 0.00000E+0, 2.00000E-2, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000, -9999.000,   1,1.0000
    98,'1 ',   585.000,    33.013,   999.000,  -999.000,1.04000,     0,   650.
↪000, 0.00000E+0, 9.75000E-2, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000, -9999.000,   1,1.0000
   101,'1 ',  1200.000,   506.878,   999.000,  -999.000,1.05000,     0,  1350.
↪000, 0.00000E+0, 2.29500E-1, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000, -9999.000,   1,1.0000
   115,'1 ',   550.000,   269.536,   999.000,  -999.000,1.02000,     0,   100.
↪000, 0.00000E+0, 2.00000E-2, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000, -9999.000,   1,1.0000
   119,'1 ',   550.000,   145.624,   999.000,  -999.000,1.00000,     0,   100.
↪000, 0.00000E+0, 2.00000E-2, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000, -9999.000,   1,1.0000
   120,'1 ',   700.000,    57.157,   999.000,  -999.000,1.02000,     0,   100.
↪000, 0.00000E+0, 2.00000E-2, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000, -9999.000,   1,1.0000
   121,'1 ',   145.000,   157.850,   999.000,  -999.000,1.04000,     0,   100.
↪000, 0.00000E+0, 2.00000E-2, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000, -9999.000,   1,1.0000
   122,'1 ',   155.000,   127.774,   999.000,  -999.000,1.05000,     0,   100.
↪000, 0.00000E+0, 2.00000E-2, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000, -9999.000,   1,1.0000
   123,'1 ',   655.000,    58.623,   999.000,  -999.000,1.01000,     0,   100.
↪000, 0.00000E+0, 2.00000E-2, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000, -9999.000,   1,1.0000
   130,'1 ',   490.000,  -467.860,   999.000,  -999.000,1.02000,     0,   100.
↪000, 0.00000E+0, 2.00000E-2, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000, -9999.000,   1,1.0000
   133,'1 ',  1700.000,  -141.013,   999.000,  -999.000,1.02000,     0,   100.
↪000, 0.00000E+0, 2.00000E-2, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000, -9999.000,   1,1.0000
   134,'1 ',   880.000,    43.093,   999.000,  -999.000,1.03000,     0,   100.
↪000, 0.00000E+0, 2.00000E-2, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000, -9999.000,   1,1.0000
   135,'1 ',  2330.000,  -100.279,   999.000,  -999.000,1.02000,     0,   100.
↪000, 0.00000E+0, 2.00000E-2, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000, -9999.000,   1,1.0000
   137,'1 ',    70.000,   174.600,   999.000,  -999.000,1.04000,     0,   100.
↪000, 0.00000E+0, 2.00000E-2, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000, -9999.000,   1,1.0000
   139,'1 ',   115.000,    18.439,   999.000,  -999.000,1.01000,     0,   100.
↪000, 0.00000E+0, 2.00000E-2, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000, -9999.000,   1,1.0000
 0 /End of Generator data, Begin Branch data
     1,       2,'1 ', 4.00000E-4, 4.30000E-3,   0.07000,    0.00,    0.00,   ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
     1,       4,'1 ', 4.00000E-4, 4.30000E-3,   0.07000,    0.00,    0.00,   ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
```

(continues on next page)

(continued from previous page)

```
     2,      33,'1 ', 7.00000E-4, 8.20000E-3,   0.14000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
     4,       5,'1 ', 9.00000E-4, 1.01000E-2,   0.17000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
     5,       6,'1 ', 1.80000E-3, 2.17000E-2,   0.37000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
     5,      31,'1 ', 8.00000E-4, 1.29000E-2,   0.14000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
     6,       7,'1 ', 9.00000E-4, 9.40000E-3,   0.17000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
     7,       8,'1 ', 7.00000E-4, 8.90000E-3,   0.13000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
     7,      10,'1 ', 1.60000E-3, 1.95000E-2,   0.30000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
     7,      12,'1 ', 8.00000E-4, 1.35000E-2,   0.25000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
     7,      15,'1 ', 3.00000E-4, 5.90000E-3,   0.07000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
     8,       9,'1 ', 7.00000E-4, 8.20000E-3,   0.13000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
     8,      18,'1 ', 1.30000E-3, 1.73000E-2,   0.32000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
     9,      30,'1 ', 1.10000E-3, 1.33000E-2,   0.21000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    12,      13,'1 ', 8.00000E-4, 1.40000E-2,   0.26000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    13,      14,'1 ', 6.00000E-4, 9.60000E-3,   0.18000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    14,      15,'1 ', 2.20000E-3, 3.50000E-2,   0.36000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    16,      17,'1 ', 3.20000E-3, 3.23000E-2,   0.53000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    16,      29,'1 ', 7.00000E-3, 8.60000E-3,   0.15000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    17,      18,'1 ', 1.40000E-3, 1.47000E-2,   0.24000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    17,      19,'1 ', 4.30000E-3, 4.74000E-2,   0.78000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    17,      20,'1 ', 5.70000E-3, 6.25000E-2,   1.03000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    19,      20,'1 ', 1.40000E-3, 1.51000E-2,   0.25000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    29,      30,'1 ', 1.30000E-3, 1.51000E-2,   0.26000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    29,      37,'1 ', 3.50000E-3, 4.11000E-2,   0.70000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    30,      31,'1 ', 1.30000E-3, 2.13000E-2,   0.22000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    31,      32,'1 ', 8.00000E-4, 1.28000E-2,   0.13000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    32,      33,'1 ', 2.00000E-4, 2.60000E-3,   0.04000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
```

(continues on next page)

(continued from previous page)

```
    32,      35,'1 ', 8.00000E-4, 1.12000E-2,   0.15000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    33,      34,'1 ', 6.00000E-4, 9.20000E-3,   0.11000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    34,      35,'1 ', 4.00000E-4, 4.60000E-3,   0.08000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    35,      73,'1 ', 2.30000E-3, 3.63000E-2,   0.38000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    37,      38,'1 ', 1.60000E-3, 1.63000E-2,   0.25000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    37,      39,'1 ', 8.00000E-4, 7.40000E-3,   0.48000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    37,      40,'1 ', 9.00000E-4, 3.15000E-2,   0.06000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    37,      43,'1 ', 1.30000E-3, 1.88000E-2,   1.31000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    38,      69,'1 ', 4.30000E-3, 4.85000E-2,   0.79000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    39,      73,'1 ', 1.90000E-3, 1.83000E-2,   0.29000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    39,      73,'2 ', 1.90000E-3, 1.83000E-2,   0.29000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    40,      44,'1 ', 8.00000E-3, 5.30000E-2,   0.40000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    41,      42,'1 ', 1.80000E-3, 8.90000E-3,   0.07000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    41,      45,'1 ', 7.76000E-2, 2.24100E-1,   0.13000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    43,      44,'1 ', 0.00000E+0, 1.10000E-2,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    43,      50,'1 ', 2.50000E-3, 2.68000E-2,   0.40000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    43,      50,'2 ', 2.50000E-3, 2.68000E-2,   0.40000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    44,      45,'1 ', 0.00000E+0, 2.05000E-2,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    44,      48,'1 ', 1.50000E-2, 1.05000E-1,   0.79000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    45,      46,'1 ', 1.23500E-1, 3.54800E-1,   0.16000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    45,      51,'1 ', 6.05000E-2, 1.89600E-1,   0.08000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    45,      71,'1 ', 1.75100E-1, 5.57500E-1,   0.12000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    46,      47,'1 ', 6.32000E-2, 1.21000E-1,   0.05000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    46,      49,'1 ', 1.64000E-1, 5.78000E-1,   0.06000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    47,      48,'1 ', 0.00000E+0, 3.30000E-2,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    50,      51,'1 ', 0.00000E+0, 2.07000E-2,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
```

(continues on next page)

(continued from previous page)

```
    50,      52,'1 ', 2.00000E-3, 2.20000E-2,   1.28000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    51,      53,'1 ', 8.44000E-2, 2.16100E-1,   0.22000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    51,      72,'1 ', 5.10000E-3, 2.32000E-2,   0.30000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    52,      53,'1 ', 1.40000E-3, 3.88000E-2,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    53,      55,'1 ', 3.00000E-2, 1.07000E-1,   0.30000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    53,      65,'1 ', 3.50000E-2, 8.00000E-2,   0.05000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    54,      56,'1 ', 1.00000E-2, 8.00000E-3,   0.07000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    54,      62,'1 ', 8.00000E-3, 6.90000E-2,   0.13000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    54,     102,'1 ', 1.60000E-3, 2.48000E-2,   0.06000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    54,     103,'1 ', 1.60000E-3, 2.48000E-2,   0.06000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    55,      57,'1 ', 7.40000E-3, 4.00000E-2,   0.02000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    55,      59,'1 ', 1.60000E-2, 8.17000E-2,   0.04000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    56,      58,'1 ', 2.00000E-3, 1.50000E-2,   0.11000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    57,      59,'1 ', 1.40000E-2, 6.80000E-2,   0.03000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    58,      60,'1 ', 3.50000E-3, 3.44000E-2,   0.27000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    58,      62,'1 ', 2.00000E-3, 1.80000E-2,   0.03000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    59,      61,'1 ', 6.80000E-2, 1.43000E-1,   0.06000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    60,     140,'1 ', 3.90000E-3, 3.63000E-2,   0.07000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    62,      63,'1 ', 1.29000E-2, 8.03000E-2,   0.15000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    63,      64,'1 ', 0.00000E+0, 4.60000E-2,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    63,      67,'1 ', 1.20000E-2, 9.30000E-2,   0.18000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    64,      65,'1 ', 3.50000E-2, 8.00000E-2,   0.05000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    64,      68,'1 ', 9.88000E-2, 2.31000E-1,   0.13000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    66,      67,'1 ', 0.00000E+0, 2.31000E-2,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    66,      69,'1 ', 1.80000E-3, 2.74000E-2,   0.27000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    66,     134,'1 ', 7.60000E-4, 1.14100E-2,   1.16000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
```

(continues on next page)

(continued from previous page)

```
    67,      68,'1 ', 0.00000E+0, 4.90000E-2,    0.00000,     0.00,     0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,    0.00,    1,1.0000
    67,      70,'1 ', 7.00000E-3, 6.19000E-2,    0.12000,     0.00,     0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,    0.00,    1,1.0000
    67,     138,'1 ', 6.30000E-3, 4.91000E-2,    0.09000,     0.00,     0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,    0.00,    1,1.0000
    68,      71,'1 ', 1.20300E-1, 2.04800E-1,    0.07000,     0.00,     0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,    0.00,    1,1.0000
    69,      71,'1 ', 0.00000E+0, 3.75000E-2,    0.00000,     0.00,     0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,    0.00,    1,1.0000
    70,      71,'1 ', 0.00000E+0, 4.90000E-2,    0.00000,     0.00,     0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,    0.00,    1,1.0000
    71,      72,'1 ', 5.30000E-3, 1.35000E-2,    0.10000,     0.00,     0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,    0.00,    1,1.0000
    73,      74,'1 ', 2.20000E-3, 1.96000E-2,    0.34000,     0.00,     0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,    0.00,    1,1.0000
    73,      74,'2 ', 2.20000E-3, 1.96000E-2,    0.34000,     0.00,     0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,    0.00,    1,1.0000
    74,      77,'1 ', 3.00000E-4, 4.30000E-3,    0.08000,     0.00,     0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,    0.00,    1,1.0000
    74,      78,'1 ', 5.00000E-4, 4.50000E-3,    0.32000,     0.00,     0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,    0.00,    1,1.0000
    74,      78,'2 ', 5.00000E-4, 4.50000E-3,    0.32000,     0.00,     0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,    0.00,    1,1.0000
    75,     124,'1 ', 7.00000E-4, 1.75000E-2,    1.39000,     0.00,     0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,    0.00,    1,1.0000
    76,      77,'1 ', 3.00000E-3, 6.80000E-3,    1.37000,     0.00,     0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,    0.00,    1,1.0000
    78,      79,'1 ', 3.00000E-4, 1.53000E-2,    0.00000,     0.00,     0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,    0.00,    1,1.0000
    78,      80,'1 ', 1.35000E-2, 5.82000E-2,    0.50000,     0.00,     0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,    0.00,    1,1.0000
    78,      81,'1 ', 5.00000E-4, 2.76000E-2,    0.00000,     0.00,     0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,    0.00,    1,1.0000
    78,      82,'1 ', 5.00000E-4, 3.08000E-2,    0.00000,     0.00,     0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,    0.00,    1,1.0000
    81,     125,'1 ', 1.00000E-4, 1.10000E-3,    0.00000,     0.00,     0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,    0.00,    1,1.0000
    83,     112,'1 ', 1.85000E-2, 1.53700E-1,    0.00000,     0.00,     0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,    0.00,    1,1.0000
    83,     113,'1 ', 3.70000E-3, 2.02000E-2,    0.00000,     0.00,     0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,    0.00,    1,1.0000
    84,     116,'1 ', 1.50000E-3, 1.04500E-1,    0.00000,     0.00,     0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,    0.00,    1,1.0000
    85,      88,'1 ', 1.60000E-3, 2.02000E-2,    0.03000,     0.00,     0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,    0.00,    1,1.0000
    85,      88,'2 ', 1.60000E-3, 2.02000E-2,    0.03000,     0.00,     0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,    0.00,    1,1.0000
    85,     105,'1 ', 1.00000E-2, 1.09000E-1,    0.18000,     0.00,     0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,    0.00,    1,1.0000
    85,     105,'2 ', 1.00000E-2, 1.09000E-1,    0.18000,     0.00,     0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,    0.00,    1,1.0000
```

(continues on next page)

(continued from previous page)

```
    85,     112,'1 ', 6.90000E-3, 5.95000E-2,   0.10000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    85,     112,'2 ', 6.90000E-3, 5.95000E-2,   0.10000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    87,     115,'1 ', 1.00000E-4, 1.80000E-3,   0.02000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    88,      89,'1 ', 5.00000E-4, 2.05000E-2,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    88,     105,'1 ', 1.12000E-2, 9.93000E-2,   0.17000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    88,     105,'2 ', 1.12000E-2, 9.93000E-2,   0.17000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    89,      90,'1 ', 1.30000E-3, 4.90000E-3,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    89,     105,'1 ', 8.94000E-2, 3.36300E-1,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    89,     113,'1 ', 1.45600E-1, 4.11500E-1,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    90,     114,'1 ', 2.40000E-3, 1.50000E-2,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    91,      92,'1 ', 5.80000E-3, 9.28000E-2,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    91,      93,'1 ', 4.21000E-2, 2.93200E-1,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    91,      95,'1 ', 2.61000E-2, 1.84400E-1,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    91,      98,'1 ', 1.20000E-2, 8.54000E-2,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    91,     110,'1 ', 1.70000E-3, 1.17000E-2,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    92,      97,'1 ', 8.00000E-4, 3.03000E-2,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    93,      94,'1 ', 4.20000E-3, 2.83000E-2,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    93,      95,'1 ', 1.87000E-2, 1.28400E-1,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    94,     111,'1 ', 4.90000E-3, 3.36000E-2,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    95,      96,'1 ', 1.36000E-2, 9.55000E-2,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    96,      97,'1 ', 1.60000E-2, 7.67000E-2,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    98,      99,'1 ', 3.00000E-4, 9.20000E-3,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    99,     100,'1 ', 1.00000E-3, 7.02000E-2,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
   101,     104,'1 ', 7.70000E-3, 7.73000E-2,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
   101,     105,'1 ', 1.88000E-2, 1.49300E-1,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
   101,     107,'1 ', 7.60000E-3, 7.51000E-2,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
```

(continues on next page)

(continued from previous page)

```
  101,    108,'1 ', 7.80000E-3, 7.72000E-2,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  101,    109,'1 ', 7.80000E-3, 7.83000E-2,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  101,    110,'1 ', 4.82000E-2, 2.63600E-1,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  104,    106,'1 ', 1.80000E-3, 1.80000E-2,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  104,    110,'1 ', 6.00000E-4, 6.20000E-3,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  105,    106,'1 ', 1.65000E-2, 1.15600E-1,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  105,    107,'1 ', 1.15000E-2, 1.10300E-1,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  105,    111,'1 ', 1.31000E-2, 8.09000E-2,   0.13000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  105,    112,'1 ', 8.00000E-3, 4.97000E-2,   0.30000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  105,    113,'1 ', 3.82400E-1, 9.47900E-1,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  107,    110,'1 ', 2.10000E-3, 1.86000E-2,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  108,    110,'1 ', 2.60000E-3, 1.94000E-2,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  108,    111,'1 ', 5.90000E-3, 5.77000E-2,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  109,    110,'1 ', 2.60000E-3, 1.94000E-2,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  109,    111,'1 ', 5.90000E-3, 5.77000E-2,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  112,    113,'1 ', 1.01000E-2, 8.73000E-2,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  114,    117,'1 ', 6.63000E-2, 3.69600E-1,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  114,    119,'1 ', 8.76000E-2, 2.54000E-1,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  114,    121,'1 ', 6.30000E-3, 8.02000E-2,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  115,    116,'1 ', 7.33000E-2, 9.79900E-1,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  115,    117,'1 ', 1.00000E-3, 1.14000E-2,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  115,    118,'1 ', 1.65700E-1, 1.55100E-1,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  115,    119,'1 ', 3.30000E-3, 3.92000E-2,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  115,    121,'1 ', 2.20000E-2, 2.98400E-1,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  115,    135,'1 ', 0.00000E+0, 6.30900E-1,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  115,    139,'1 ', 0.00000E+0, 1.24300E-1,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
```

(continues on next page)

(continued from previous page)

```
  116,    117,'1 ', 4.74000E-2, 2.63100E-1,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  116,    118,'1 ', 1.17000E-2, 7.44000E-2,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  116,    120,'1 ', 2.30700E-1, 7.04100E-1,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  116,    121,'1 ', 9.51000E-2, 4.57600E-1,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  116,    122,'1 ', 3.30000E-3, 1.28700E-1,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  117,    118,'1 ', 2.00000E-3, 1.99000E-2,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  117,    119,'1 ', 1.70000E-3, 1.89000E-2,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  117,    120,'1 ', 6.70000E-3, 7.97000E-2,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  117,    121,'1 ', 1.50000E-2, 8.69000E-2,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  117,    122,'1 ', 3.83000E-2, 8.86000E-2,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  118,    120,'1 ', 5.00000E-3, 6.29000E-2,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  118,    121,'1 ', 1.39100E-1, 4.79800E-1,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  118,    122,'1 ', 9.50000E-2, 5.07100E-1,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  118,    123,'1 ', 1.40000E-3, 1.74000E-2,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  119,    120,'1 ', 1.06500E-1, 3.49800E-1,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  119,    121,'1 ', 1.63600E-1, 5.80400E-1,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  121,    122,'1 ', 1.25000E-1, 4.69900E-1,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  122,    135,'1 ', 5.73000E-2, 7.58100E-1,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  122,    139,'1 ', 0.00000E+0, 1.53200E-1,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  124,    125,'1 ', 2.50000E-3, 7.30000E-2,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  124,    126,'1 ', 0.00000E+0, 8.39000E-2,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  124,    127,'1 ', 4.00000E-4, 1.05000E-2,   0.72000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  124,    128,'1 ', 5.00000E-4, 1.17000E-2,   0.84000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  125,    126,'1 ', 0.00000E+0, 4.11000E-2,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  125,    128,'1 ', 4.90000E-3, 2.63400E-1,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  126,    127,'1 ', 0.00000E+0, 2.53200E-1,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
```

(continues on next page)

(continued from previous page)

```
   126,    128,'1 ', 0.00000E+0, 2.74100E-1,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
   126,    138,'1 ', 0.00000E+0, 6.80500E-1,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
   127,    128,'1 ', 0.00000E+0, 1.33900E-1,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
   127,    132,'1 ', 9.00000E-4, 2.21000E-2,   1.62000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
   128,    129,'1 ', 0.00000E+0, 1.38100E-1,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
   128,    130,'1 ', 7.00000E-4, 1.75000E-2,   1.26000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
   129,    130,'1 ', 4.00000E-4, 8.50000E-3,   0.60000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
   130,    131,'1 ', 2.00000E-4, 3.80000E-3,   0.54000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
   130,    132,'1 ', 7.00000E-4, 1.73000E-2,   1.25000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
   131,    133,'1 ', 2.00000E-3, 3.90000E-2,   2.77000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
   131,    135,'1 ', 5.70000E-3, 6.99000E-2,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
   131,    139,'1 ', 5.90000E-3, 1.05200E-1,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
   132,    133,'1 ', 1.20000E-3, 2.93000E-2,   2.09000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
   132,    134,'1 ', 2.35000E-3, 4.38800E-2,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
   132,    135,'1 ', 1.20000E-3, 2.88000E-2,   2.06000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
   132,    138,'1 ', 3.31000E-2, 4.22400E-1,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
   133,    135,'1 ', 3.00000E-4, 6.30000E-3,   0.45000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
   134,    135,'1 ', 4.60000E-3, 6.25000E-2,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
   134,    136,'1 ', 4.60000E-3, 7.82000E-2,   0.79000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
   135,    139,'1 ', 1.77000E-2, 6.61000E-2,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
   136,    139,'1 ', 0.00000E+0, 2.05400E-1,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
   137,    138,'1 ', 2.55300E-1, 7.18000E-1,   0.00000,    0.00,    0.00,
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
 0 /End of Branch data, Begin Transformer data
     1,    21,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'TWO-WINDINGS',1,
↪1,1.0000
 0.00000E+0, 2.00000E-2,   100.00
1.00000,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.10000,
↪0.90000, 1.10000, 0.90000,  33, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
     3,     2,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'TWO-WINDINGS',1,
```

(continues on next page)

(continued from previous page)

```
↪1,1.0000
 1.60000E-3, 4.35000E-2,   100.00
1.00000,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.10000,␣
↪0.90000, 1.10000, 0.90000,  33, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
     3,     4,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'TWO-WINDINGS',1,  ␣
↪1,1.0000
 1.60000E-3, 4.35000E-2,   100.00
1.00000,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.10000,␣
↪0.90000, 1.10000, 0.90000,  33, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
    10,    11,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'TWO-WINDINGS',1,  ␣
↪1,1.0000
 7.00000E-4, 1.38000E-2,   100.00
1.00000,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.10000,␣
↪0.90000, 1.10000, 0.90000,  33, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
    10,    22,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'TWO-WINDINGS',1,  ␣
↪1,1.0000
 7.00000E-4, 1.42000E-2,   100.00
1.00000,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.10000,␣
↪0.90000, 1.10000, 0.90000,  33, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
    11,    23,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'TWO-WINDINGS',1,  ␣
↪1,1.0000
 9.00000E-4, 1.80000E-2,   100.00
1.00000,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.10000,␣
↪0.90000, 1.10000, 0.90000,  33, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
    13,    24,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'TWO-WINDINGS',1,  ␣
↪1,1.0000
 0.00000E+0, 1.43000E-2,   100.00
1.00000,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.10000,␣
↪0.90000, 1.10000, 0.90000,  33, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
    14,    25,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'TWO-WINDINGS',1,  ␣
↪1,1.0000
 5.00000E-4, 2.72000E-2,   100.00
1.00000,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.10000,␣
↪0.90000, 1.10000, 0.90000,  33, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
    16,    27,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'TWO-WINDINGS',1,  ␣
↪1,1.0000
 6.00000E-4, 2.32000E-2,   100.00
1.00000,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.10000,␣
↪0.90000, 1.10000, 0.90000,  33, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
    20,    26,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'TWO-WINDINGS',1,  ␣
↪1,1.0000
 8.00000E-4, 1.56000E-2,   100.00
1.00000,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.10000,␣
↪0.90000, 1.10000, 0.90000,  33, 0, 0.00000, 0.00000,  0.000
```

(continues on next page)

(continued from previous page)

```
1.00000,   0.000
    28,    29,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'TWO-WINDINGS',1,  ␣
↪1,1.0000
 0.00000E+0, 1.81000E-2,   100.00
1.00000,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.10000,␣
↪0.90000, 1.10000, 0.90000,  33, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
    33,    36,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'TWO-WINDINGS',1,  ␣
↪1,1.0000
 0.00000E+0, 2.50000E-2,   100.00
1.00000,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.10000,␣
↪0.90000, 1.10000, 0.90000,  33, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
    40,    41,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'TWO-WINDINGS',1,  ␣
↪1,1.0000
 0.00000E+0, 1.61000E-2,   100.00
1.00000,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.10000,␣
↪0.90000, 1.10000, 0.90000,  33, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
    49,    48,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'TWO-WINDINGS',1,  ␣
↪1,1.0000
 1.56000E-2, 1.53600E-1,   100.00
1.00000,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.10000,␣
↪0.90000, 1.10000, 0.90000,  33, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
    48,   100,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'TWO-WINDINGS',1,  ␣
↪1,1.0000
 1.00000E-3, 1.59500E-1,   100.00
1.00000,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.10000,␣
↪0.90000, 1.10000, 0.90000,  33, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
    54,    52,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'TWO-WINDINGS',1,  ␣
↪1,1.0000
 2.00000E-3, 2.33000E-2,   100.00
1.00000,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.10000,␣
↪0.90000, 1.10000, 0.90000,  33, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
    54,    55,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'TWO-WINDINGS',1,  ␣
↪1,1.0000
 0.00000E+0, 1.30000E-2,   100.00
1.00000,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.10000,␣
↪0.90000, 1.10000, 0.90000,  33, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
    56,    57,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'TWO-WINDINGS',1,  ␣
↪1,1.0000
 0.00000E+0, 1.42000E-1,   100.00
1.00000,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.10000,␣
↪0.90000, 1.10000, 0.90000,  33, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
    58,    59,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'TWO-WINDINGS',1,  ␣
↪1,1.0000
 0.00000E+0, 9.30000E-3,   100.00
```

(continues on next page)

(continued from previous page)

```
1.00000,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.10000,␣
↪0.90000, 1.10000, 0.90000,  33, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
    60,    61,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'TWO-WINDINGS',1,  ␣
↪1,1.0000
 0.00000E+0, 6.40000E-2,   100.00
1.00000,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.10000,␣
↪0.90000, 1.10000, 0.90000,  33, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
    75,    76,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'TWO-WINDINGS',1,  ␣
↪1,1.0000
 1.00000E-4, 7.40000E-3,   100.00
1.00000,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.10000,␣
↪0.90000, 1.10000, 0.90000,  33, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
    84,    83,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'TWO-WINDINGS',1,  ␣
↪1,1.0000
 5.20000E-3, 1.74000E-2,   100.00
1.00000,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.10000,␣
↪0.90000, 1.10000, 0.90000,  33, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
    85,    86,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'TWO-WINDINGS',1,  ␣
↪1,1.0000
 0.00000E+0, 5.70000E-3,   100.00
1.00000,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.10000,␣
↪0.90000, 1.10000, 0.90000,  33, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
    87,    85,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'TWO-WINDINGS',1,  ␣
↪1,1.0000
 2.00000E-4, 2.46000E-2,   100.00
1.00000,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.10000,␣
↪0.90000, 1.10000, 0.90000,  33, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
   101,   102,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'TWO-WINDINGS',1,  ␣
↪1,1.0000
 7.00000E-4, 2.11000E-2,   100.00
1.00000,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.10000,␣
↪0.90000, 1.10000, 0.90000,  33, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
   101,   103,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'TWO-WINDINGS',1,  ␣
↪1,1.0000
 6.00000E-4, 1.51000E-2,   100.00
1.00000,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.10000,␣
↪0.90000, 1.10000, 0.90000,  33, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
   137,   136,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'TWO-WINDINGS',1,  ␣
↪1,1.0000
 8.00000E-4, 2.39000E-2,   100.00
1.00000,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.10000,␣
↪0.90000, 1.10000, 0.90000,  33, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
 0 /End of Transformer data, Begin Area interchange data
```

(continues on next page)

(continued from previous page)

```
   1,     0,     0.000,    10.000,'NEPOOL      '
   2,     0,     0.000,    10.000,'NYISO       '
   3,     0,     0.000,    10.000,'NYISO2      '
   4,     0,     0.000,    10.000,'IESO        '
   5,     0,     0.000,    10.000,'MISO        '
   6,     0,     0.000,    10.000,'PJM         '
 0 /End of Area interchange data, Begin Two-terminal dc line data
 0 /End of Two-terminal dc line data, Begin VSC dc line data
 0 /End of VSC dc line data, Begin Impedance correction table data
 0 /End of Impedance correction table data, Begin Multi-terminal dc line data
 0 /End of Multi-terminal dc line data, Begin Multi-section line data
 0 /End of Multi-section line data, Begin Zone data
 0 /End of Zone data, Begin Inter-area transfer data
 0 /End of Inter-area transfer data, Begin Owner data
 0 /End of Owner data, Begin FACTS device data
 0 /End of FACTS device data, Begin Switched shunt data
 0 /End of Switched shunt data, Begin GNE device data
 0 /End of GNE device data
Q
```

Dynamic Data (PSS/E DYR format)

```
data = !cat npcc.dyr
print('\n'.join(data))
```

```
     21 'GENROU' 1     5.7000      0.30000E-01  0.35000      0.50000E-01
          4.6400       0.0000       1.9050       1.8075      0.36000
         0.36000      0.23270      0.20270       0.0000       0.0000    /
     22 'GENROU' 1     5.6900      0.30000E-01  0.35000      0.50000E-01
          4.0857       0.0000       1.8340       1.8060      0.30520
         0.30520      0.220955      0.200955       0.0000       0.0000    /
     23 'GENROU' 1     5.6000      0.30000E-01  0.35000      0.50000E-01
          2.4467       0.0000       1.7400       1.6800      0.24450
         0.24450      0.100799     0.090799   0.0000       0.0000    /
     23 'GENROU' 2     5.2000      0.30000E-01  0.35000      0.50000E-01
          6.2000       0.0000       2.1150       2.0400      0.54600
         0.54600      0.225511      0.205511       0.0000       0.0000    /
     24 'GENROU' 1     7.3000      0.30000E-01  0.35000      0.50000E-01
          4.3500       0.0000       2.0320       1.9280      0.38400
         0.38400      0.23780      0.20780       0.0000       0.0000    /
     25 'GENROU' 1     5.7000      0.30000E-01  0.35000      0.50000E-01
          4.0615       0.0000       1.9175       1.8980      0.31200
         0.31200      0.20930      0.19930       0.0000       0.0000    /
     26 'GENROU' 1     4.7900      0.30000E-01  0.35000      0.50000E-01
          3.8000       0.0000       1.8900       1.8450      0.52200
         0.52200      0.28874      0.20874       0.0000       0.0000    /
     27 'GENROU' 1     6.7000      0.30000E-01  0.35000      0.50000E-01
          4.0500       0.0000       1.7400       1.6800      0.33000
         0.33000      0.18450      0.10450       0.0000       0.0000    /
```

(continues on next page)

(continued from previous page)

```
    36 'GENROU' 1     5.7000      0.30000E-01  0.35000      0.50000E-01
         4.6615       0.0000       1.9175       1.8590      0.52650
        0.52650      0.319930      0.30993       0.0000       0.0000    /
    42 'GENROU' 1     5.0000      0.30000E-01  0.35000      0.50000E-01
         4.7150       0.0000       1.2648       1.2108      0.23800
        0.23800      0.17800      0.10800       0.0000       0.0000    /
    47 'GENROU' 1     5.0000      0.30000E-01  0.35000      0.50000E-01
         3.3711       0.0000       1.3410      0.76950      0.49500
        0.49500      0.35190      0.30190       0.0000       0.0000    /
    48 'GENROU' 1     5.0000      0.30000E-01  0.35000      0.50000E-01
         3.0340       0.0000       1.4900      0.85500      0.55000
        0.55000      0.39100      0.30100       0.0000       0.0000    /
    50 'GENROU' 1     6.7000      0.30000E-01  0.35000      0.50000E-01
         4.9200       0.0000       1.3750       1.3050      0.39000
        0.39000      0.29500      0.20500       0.0000       0.0000    /
    51 'GENROU' 1     5.5000      0.30000E-01  0.35000      0.50000E-01
         14.307       0.0000      0.46545      0.44115      0.93750E-01
        0.93750E-01  0.61950E-01  0.60950E-01   0.0000       0.0000    /
    53 'GENCLS' 1     37.000       37.000    /
    54 'GENROU' 1     7.0000      0.30000E-01  0.35000      0.50000E-01
         7.8831       0.0000      0.85930      0.58500      0.36790
        0.36790      0.32500      0.30500       0.0000       0.0000    /
    54 'GENROU' 2     7.0000      0.30000E-01  0.35000      0.50000E-01
         7.8831       0.0000      0.85930      0.58500      0.36790
        0.36790      0.32500      0.30500       0.0000       0.0000    /
    55 'GENROU' 1     7.0000      0.30000E-01  0.35000      0.50000E-01
         4.9817       0.0000       1.2192      0.77160      0.43680
        0.43680      0.36240      0.30240       0.0000       0.0000    /
    56 'GENROU' 1     5.5000      0.30000E-01  0.35000      0.50000E-01
         4.3880       0.0000       1.9235       1.8485      0.35750
        0.35750      0.25750      0.20750       0.0000       0.0000    /
    57 'GENROU' 1     5.6000      0.30000E-01  0.35000      0.50000E-01
         5.0600       0.0000       1.1805       1.1294      0.24465
        0.24465      0.16240      0.10240       0.0000       0.0000    /
    60 'GENROU' 1     5.5000      0.30000E-01  0.35000      0.50000E-01
         4.8311       0.0000       1.7334       1.6673      0.32490
        0.32490      0.23535      0.20535       0.0000       0.0000    /
    61 'GENROU' 1     5.0000      0.30000E-01  0.35000      0.50000E-01
         4.1600       0.0000       1.2768       1.2228      0.24960
        0.24960      0.18940      0.10940       0.0000       0.0000    /
    65 'GENCLS' 1     10.710       10.710    /
    68 'GENCLS' 1     3.7700       3.7700    /
    71 'GENCLS' 1     8.5000       8.5000    /
    72 'GENCLS' 1     11.770       11.770    /
    78 'GENCLS' 1     1000.0       1000.0    /
    79 'GENROU' 1     8.0000      0.30000E-01  0.35000      0.50000E-01
         4.1739       0.0000       2.0010       1.9550      0.36800
        0.36800      0.32200      0.30200       0.0000       0.0000    /
    80 'GENROU' 1     4.0000      0.30000E-01  0.35000      0.50000E-01
         2.9750       0.0000       1.7040       1.5440      0.37600
        0.37600      0.20000      0.19000       0.0000       0.0000    /
    82 'GENROU' 1     5.9000      0.30000E-01  0.35000      0.50000E-01
```

(continues on next page)

(continued from previous page)

```
      3.9200       0.0000       1.5450       1.5100       0.31000
     0.31000      0.25000      0.20000       0.0000       0.0000    /
  86 'GENROU' 1     6.0000      0.30000E-01  0.35000      0.50000E-01
      4.1579       0.0000       1.9000       1.5060       0.38000
     0.38000       0.2109      0.20090       0.0000       0.0000    /
  98 'GENROU' 1     4.6000      0.30000E-01  0.35000      0.50000E-01
      11.077       0.0000       0.4550      0.32500       0.143
      0.143        0.0975       0.0775       0.0000       0.0000    /
 101 'GENROU' 1     5.5000      0.30000E-01  0.35000      0.50000E-01
      4.0740       0.0000       1.0125       0.648       0.351
     0.35100       0.2295       0.2095       0.0000       0.0000    /
  91 'GENCLS' 1     98.700       98.700    /
  92 'GENCLS' 1     27.000       27.000    /
  97 'GENCLS' 1     36.000       36.000    /
 115 'GENCLS' 1     46.000       46.000    /
 119 'GENCLS' 1     30.000       30.000    /
 120 'GENCLS' 1     1000.0       1000.0    /
 121 'GENCLS' 1     66.000       66.000    /
 122 'GENCLS' 1     190.00       190.00    /
 123 'GENCLS' 1     33.000       33.000    /
 130 'GENCLS' 1     24.000       24.000    /
 133 'GENCLS' 1     1000.0       1000.0    /
 134 'GENCLS' 1     44.000       44.000    /
 135 'GENCLS' 1     115.00       115.00    /
 137 'GENCLS' 1     3.5000       3.5000    /
 139 'GENCLS' 1     1000.0       1000.0    /

  21 'TGOV1'  1    0.30000E-01  0.50000       1.0000      0.30000
      6.0000       6.0000       0.0000    /
  22 'TGOV1'  1    0.30000E-01  0.50000       1.0000      0.30000
      6.0000       6.0000       0.0000    /
  23 'TGOV1'  1    0.30000E-01  0.50000       1.0000      0.30000
      6.0000       6.0000       0.0000    /
  23 'TGOV1'  2    0.30000E-01  0.50000       1.0000      0.30000
      6.0000       6.0000       0.0000    /
  24 'TGOV1'  1    0.30000E-01  0.50000       1.0000      0.30000
      6.0000       6.0000       0.0000    /
  25 'TGOV1'  1    0.30000E-01  0.50000       1.0000      0.30000
      6.0000       6.0000       0.0000    /
  26 'TGOV1'  1    0.30000E-01  0.50000       1.0000      0.30000
      6.0000       6.0000       0.0000    /
  27 'TGOV1'  1    0.30000E-01  0.50000       1.0000      0.30000
      6.0000       6.0000       0.0000    /
  36 'TGOV1'  1    0.50000E-01  10.0000       1.0000      0.30000
      6.0000       6.0000       0.0000    /
  42 'TGOV1'  1    0.30000E-01  0.50000       1.0000      0.30000
      6.0000       6.0000       0.0000    /
  47 'TGOV1'  1    0.30000E-01  0.50000       1.0000      0.30000
      6.0000       6.0000       0.0000    /
  48 'TGOV1'  1    0.30000E-01  0.50000       1.0000      0.30000
      6.0000       6.0000       0.0000    /
  50 'TGOV1'  1    0.30000E-01  0.50000       1.0000      0.30000
```

(continues on next page)

(continued from previous page)

```
     6.0000       6.0000       0.0000    /
51 'TGOV1'  1    0.30000E-01  0.50000       1.0000      0.30000
     6.0000       6.0000       0.0000    /
54 'TGOV1'  1    0.30000E-01  0.50000       1.0000      0.30000
     6.0000       6.0000       0.0000    /
54 'TGOV1'  2    0.30000E-01  0.50000       1.0000      0.30000
     6.0000       6.0000       0.0000    /
55 'TGOV1'  1    0.30000E-01  0.50000       1.0000      0.30000
     6.0000       6.0000       0.0000    /
56 'TGOV1'  1    0.30000E-01  0.50000       1.0000      0.30000
     6.0000       6.0000       0.0000    /
57 'TGOV1'  1    0.30000E-01  0.50000       1.0000      0.30000
     6.0000       6.0000       0.0000    /
60 'TGOV1'  1    0.30000E-01  0.50000       1.0000      0.30000
     6.0000       6.0000       0.0000    /
61 'TGOV1'  1    0.30000E-01  0.50000       1.0000      0.30000
     6.0000       6.0000       0.0000    /
79 'TGOV1'  1    0.50000E-01  10.0000       1.0000      0.30000
     6.0000       6.0000       0.0000    /
80 'TGOV1'  1    0.30000E-01  0.50000       1.0000      0.30000
     6.0000       6.0000       0.0000    /
82 'TGOV1'  1    0.30000E-01  0.50000       1.0000      0.30000
     6.0000       6.0000       0.0000    /
86 'TGOV1'  1    0.50000E-01  10.0000       1.0000      0.30000
     6.0000       6.0000       0.0000    /
98 'TGOV1'  1    0.30000E-01  0.50000       1.0000      0.30000
     6.0000       6.0000       0.0000    /
101 'TGOV1'  1    0.30000E-01  0.50000       1.0000      0.30000
     6.0000       6.0000       0.0000    /
119 'TGOV1'  1    0.50000E-01  10.0000       100.0000      0.30000
     6.0000       6.0000       0.0000    /
133 'TGOV1'  1    0.50000E-01  10.0000       100.0000      0.30000
     6.0000       6.0000       0.0000    /

21 'IEEEX1' 1     0.0000       50.000      0.60000E-01   0.0000
     0.0000       1.0000      -1.0000     -0.20000E-01  0.50000
    0.80000E-01   1.0000       0.0000       2.0000      0.16000E-02
     3.0000       1.7300    /
22 'IEEEX1' 1     0.0000       400.00      0.20000E-01   0.0000
     0.0000       7.3000      -7.3000       1.0000      0.79000
    0.30000E-01   1.0000       0.0000       2.0000      0.16000E-02
     3.0000       1.4500    /
23 'IEEEX1' 1     0.0000       50.000      0.60000E-01   0.0000
     0.0000       1.0000      -1.0000     -0.50000E-01  0.50000
    0.80000E-01   1.0000       0.0000       2.0000      0.16000E-02
     3.0000       1.7300    /
23 'IEEEX1' 2     0.0000       50.000      0.60000E-01   0.0000
     0.0000       1.0000      -1.0000     -0.50000E-01  0.50000
    0.80000E-01   1.0000       0.0000       2.0000      0.16000E-02
     3.0000       1.7300    /
24 'IEEEX1' 1     0.0000       50.000      0.20000E-01   0.0000
     0.0000       1.0000      -1.0000     -0.40000E-01  0.47000
```

(continues on next page)

(continued from previous page)

```
    0.60000E-01   1.2500       0.0000       2.0000      0.16000E-02
     3.0000       1.7300    /
25 'IEEEX1' 1     0.0000       400.00      0.20000E-01   0.0000
     0.0000       6.5000      -6.5000       1.0000      0.73000
    0.30000E-01   1.0000       0.0000       2.0000      0.39000E-02
     3.0000       1.5550    /
26 'IEEEX1' 1     0.0000       200.00      0.20000E-01   0.0000
     0.0000       4.2600      -4.2600       1.0000       1.4000
    0.30000E-01   1.0000       0.0000       2.0000      0.16000E-02
     3.0000       1.4650    /
27 'IEEEX1' 1     0.0000       50.000      0.20000E-01   0.0000
     0.0000       1.0000      -1.0000     -0.50000E-01  0.53000
    0.33200E-01   1.2600       0.0000       2.0000      0.16000E-02
     3.0000       1.4650    /
36 'IEEEX1' 1     0.0000       62.000      0.50000E-01   0.0000
     0.0000       1.0000      -1.0000     -0.50000E-01  0.40500
    0.11400      0.50000       0.0000       2.0000      0.16000E-02
     3.0000       1.7300    /
42 'IEEEX1' 1     0.0000       25.000      0.20000       0.0000
     0.0000       1.0000      -1.0000     -0.62000E-01  0.50000
    0.65710      0.35000       0.0000       2.0000      0.16000E-02
     3.0000       2.0250    /
47 'IEEEX1' 1     0.0000       16.500      0.20000       0.0000
     0.0000       1.0000      -1.0000     -0.37000E-01  0.27000
    0.60000E-01   1.0000       0.0000       2.0000      0.58000E-02
     3.0000       1.1500    /
48 'IEEEX1' 1     0.0000       16.500      0.20000       0.0000
     0.0000       1.0000      -1.0000     -0.37000E-01  0.27000
    0.60000E-01   1.0000       0.0000       2.0000      0.58000E-02
     3.0000       1.1500    /
50 'IEEEX1' 1     0.0000       50.000      0.60000E-01   0.0000
     0.0000       1.0000      -1.0000     -0.46000E-01  0.50000
    0.80000E-01   1.0000       0.0000       2.0000      0.16000E-02
     3.0000       1.5250    /
51 'IEEEX1' 1     0.0000       23.100      0.20000       0.0000
     0.0000       1.0000      -1.0000     -0.44000E-01  0.50000
    0.71430E-01  0.35000       0.0000       2.0000      0.16000E-02
     3.0000       1.4500    /
54 'IEEEX1' 1     0.0000       50.000      0.50000E-01   0.0000
     0.0000       5.0000      -5.0000       1.0000      0.40000
    0.80000E-01   1.0000       0.0000       2.0000      0.16000E-02
     3.0000       1.4500    /
54 'IEEEX1' 2     0.0000       50.000      0.50000E-01   0.0000
     0.0000       5.0000      -5.0000       1.0000      0.40000
    0.80000E-01   1.0000       0.0000       2.0000      0.16000E-02
     3.0000       1.4500    /
55 'IEEEX1' 1     0.0000       23.100      0.20000       0.0000
     0.0000       1.0000      -1.0000     -0.50000E-01  0.50000
    0.71430E-01  0.35000       0.0000       2.0000      0.16000E-02
     3.0000       1.4500    /
56 'IEEEX1' 1     0.0000       25.000      0.20000       0.0000
     0.0000       1.0000      -1.0000     -0.50000E-01  0.50000
```

(continues on next page)

(continued from previous page)

```
        0.65710E-01  0.35000       0.0000       2.0000      0.16000E-02
         3.0000       1.6670    /
    57 'IEEEX1' 1     0.0000       25.000      0.20000       0.0000
         0.0000       1.0000      -1.0000     -0.50000E-01  0.50000
        0.65710E-01  0.35000       0.0000       2.0000      0.16000E-02
         3.0000       2.0600    /
    60 'IEEEX1' 1     0.0000       25.000      0.20000       0.0000
         0.0000       1.0000      -1.0000     -0.50000E-01  0.50000
        0.65710      0.35000       0.0000       2.0000      0.16000E-02
         3.0000       1.6670    /
    61 'IEEEX1' 1     0.0000       25.000      0.20000       0.0000
         0.0000       1.0000      -1.0000     -0.62000E-01  0.50000
        0.65710      0.35000       0.0000       2.0000      0.16000E-02
         3.0000       2.0250    /
    79 'IEEEX1' 1     0.0000       50.000      0.60000E-01   0.0000
         0.0000       1.0000      -1.0000     -0.40000E-01  0.50000
        0.80000E-01   1.0000       0.0000       2.0000      0.16000E-02
         3.0000       1.4500    /
    80 'IEEEX1' 1     0.0000       25.000      0.20000       0.0000
         0.0000       1.0000      -1.0000     -0.50000E-01  0.50000
        0.65140      0.35000       0.0000       2.0000      0.16000E-02
         3.0000       1.4500    /
    82 'IEEEX1' 1     0.0000       50.000      0.60000E-01   0.0000
         0.0000       1.0000      -1.0000     -0.40000E-01  0.50000
        0.80000E-01   1.0000       0.0000       2.0000      0.16000E-02
         3.0000       1.4500    /
```

6.4.3 CURENT WECC Verification

Prepared by Hantao Cui. Last revised 23 May 2020.

Background

The CURENT WECC system contains 179 buses and 263 branches. Two line trip scenarios are created to verify ANDES simulation results with DSATools TSAT and Siemens PTI PSS/E.

Dynamic data is based on the CURENT WECC 179-Bus test system, which uses models GENROU, TGOV1, IEEEG1, EXST2, EXDC2, ESST3A, IEEEST, ST2CUT and ESDC2A, the saturation of which could be implemented differently across software.

Simulation Parameters

Integretion method: Trapezidal Rule (ANDES and TSAT), the default second order Adams-Bashforth method (AB-2) (PSS/E)

Time step size: `1/120` sec. (Note: step size between `1/30` to `1/120` has little impact on the ANDES results. One can use `tstep=1/30` to obtain almost the same results.)

Load conversion: static loads are converted to 100% constant impedances for both P and Q.

TSAT automatic parameter correction is disabled.

Initialization

Power flow solutions are identical across all the three software.

GENROU initialization (E_{FD}, E_{TERM}, P, Q, δ, I_d and I_q) is identical to that from **PSS/E** for all cases (with and without generator saturation). Note that I_d and I_q are in machine base in PSS/E but in system base in ANDES.

GENROU initialization (including all the internal variables E'_d, E'_q, ψ_{kd}, ψ_{kq}, ψ''_d, and ψ''_q) is identical to that from **OpenIPSL**.

No controller limit violation occurs during initialization.

Conclusion

The finding is that, **for particular disturbances in large systems, neither two of ANDES, TSAT and PSS/E could match**. Implementation details in commercial software are lacking. Thus, it could be futile to insist on obtaining the same results for large systems, especially with several complex models.

Nevertheless, *Scenario 2* shows that ANDES results are trustworthy. **In fact, ANDES is open-source with all the implemented equations clearly documented, which makes the results highly credible for research and education purposes.**

```
import andes
import numpy as np
from andes.utils.tsat import tsat_to_df, psse_to_df, plot_comparison, run_cmp

andes.config_logger(stream_level=30)
```

Scenario 1: Line 38-39 trips and reconnects after 0.1 sec.

Line 38-39 is a transformer branch that connects generator #11 on Bus 39 to Bus 38. It is the only link to Bus 39.

```
ss = run_cmp('wecc.raw', dyr='wecc.dyr', fault_line='Line_221',
             t1=1.0, t2=1.1, tstep=1/120)
```

```
<Toggle Toggle_1>: Line.Line_221 status changed to 0.0 at t=1.0 sec.
<Toggle Toggle_2>: Line.Line_221 status changed to 1.0 at t=1.1 sec.
100%|██████████████████████████████████| 100/100 [00:16<00:00,  5.89%/s]
```

```
# Line_221 information
ss.Line.as_df(vin=True).iloc[220]
```

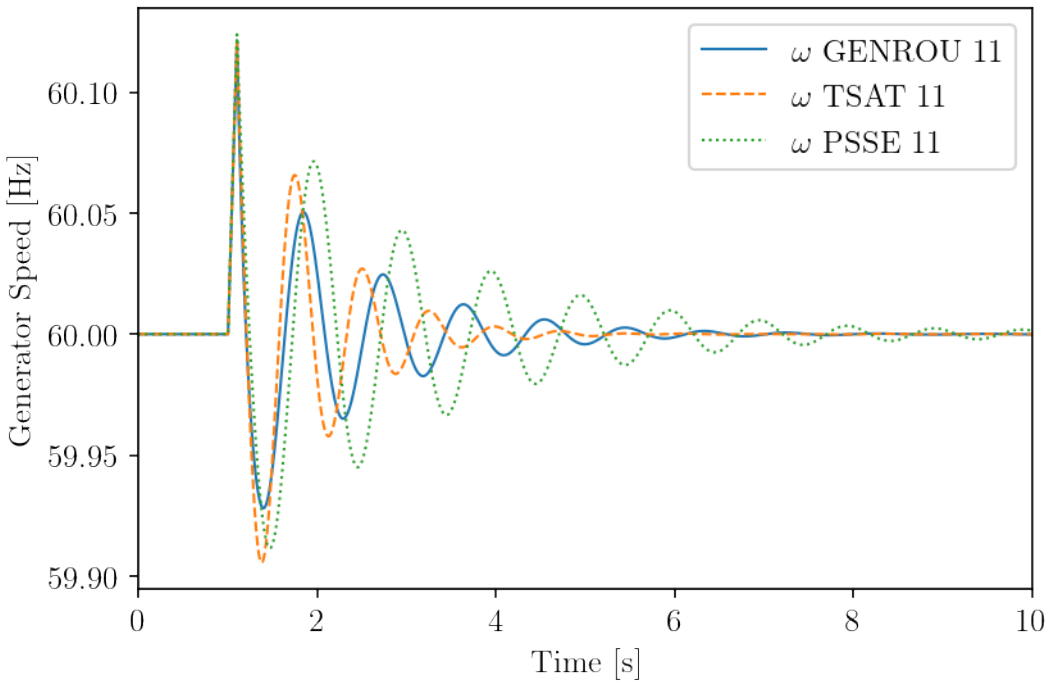
```
# Load TSAT and PSSE Outputs
omega2 = tsat_to_df('omega2.xls')
v2 = tsat_to_df('v2.xls')

omega2_psse = psse_to_df('omega2_psse.xlsx')
v2_psse = psse_to_df('v2_psse.xlsx')
omega2_psse.iloc[:, 1:] += 1
omega2_psse.iloc[:, 1:] *= 60

# prepare variable headers for TSAT and PSSE
def header_replace(fname, idx, offset, old_str, new_str):
    """
    Utility function for replacing strings in variable header
    """
    old_headers = [fname[i + offset] for i in idx]
    return [i.replace(old_str, new_str) for i in old_headers]

fname = ss.TDS.plt._fname
tsat_omega_headers = header_replace(fname, ss.GENROU.omega.a, 1, "GENROU",
↪"TSAT")
tsat_v_headers = header_replace(fname, ss.GENROU.v.a, 1 + ss.dae.n, "Bus",
↪"TSAT")

psse_omega_headers = header_replace(fname, ss.GENROU.omega.a, 1, "GENROU",
↪"PSSE")
psse_v_headers = header_replace(fname, ss.GENROU.v.a, 1 + ss.dae.n, "Bus",
↪"PSSE")
```

```
fig, ax = plot_comparison(ss, ss.GENROU.omega, omega2,
                          a=(10,), a_tsat=[10], a_psse=[10],
                          ylabel="Generator Speed [Hz]",
                          tsat_header=tsat_omega_headers,
                          psse_data = omega2_psse,
                          psse_header = psse_omega_headers,
                          scale=60, right=10,
                          legend=True,
```

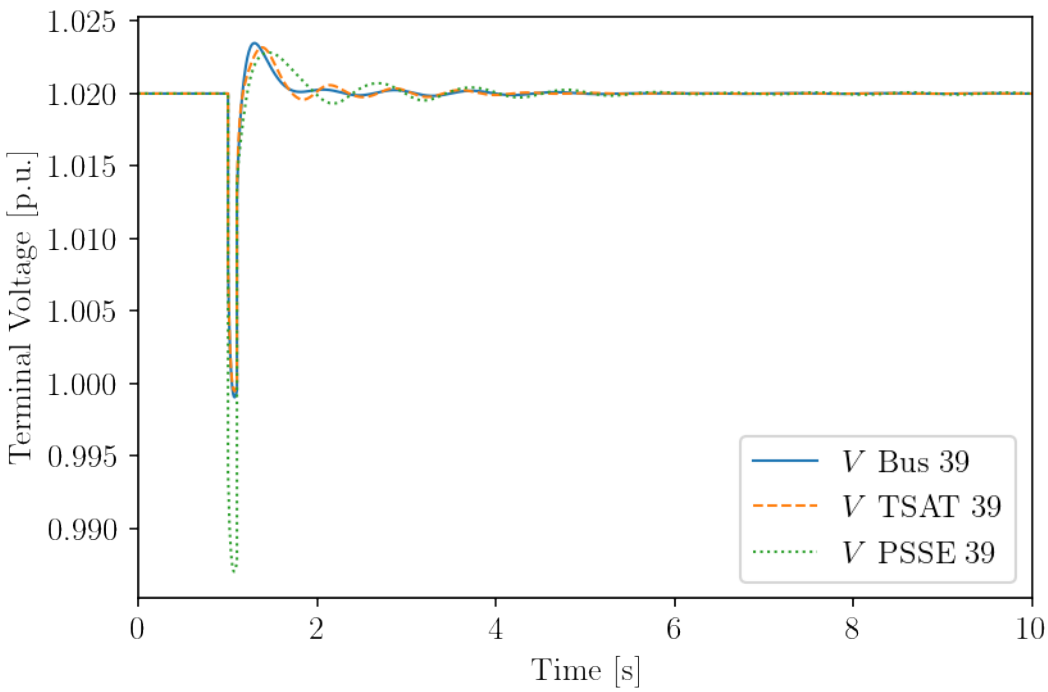
(continues on next page)

(continued from previous page)

```
                                show=False,
                               )
```




```
fig, ax = plot_comparison(ss, ss.GENROU.v, v2,
                          a=(10,), a_tsat=[10], a_psse=[10],
                          ylabel="Terminal Voltage [p.u.]",
                          tsat_header=tsat_v_headers,
                          psse_data = v2_psse,
                          psse_header = psse_v_headers,
                          scale=1, right=10,
                          legend=True,
                          show=False
                         )
```



Scenario 2: Line 4-16 trips and reconnects after 0.1 sec.

Line 4-16 is in the eastern zone that connects two generator buses (CRAIG and SAN JUAN).

```
ss2 = run_cmp('wecc.raw', dyr='wecc.dyr', fault_line='Line_2',
              t1=1.0, t2=1.1, tstep=1/120)
```

```
<Toggle Toggle_1>: Line.Line_2 status changed to 0.0 at t=1.0 sec.
<Toggle Toggle_2>: Line.Line_2 status changed to 1.0 at t=1.1 sec.
100%|██████████████████████████████████| 100/100 [00:18<00:00,  5.45%/s]
```

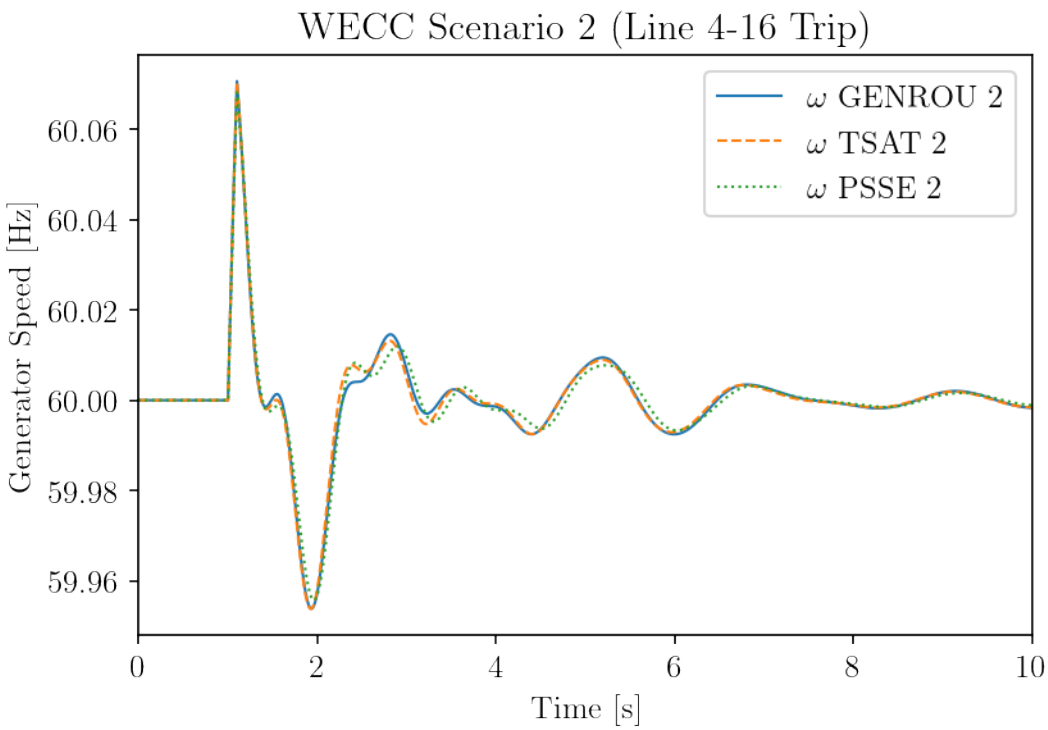
```
# Line 2 information
ss2.Line.as_df(vin=True).iloc[1]
```

```
# load data
omega = tsat_to_df('omega.xls')
v = tsat_to_df('v.xls')

omega_psse = psse_to_df('omega_psse.xlsx')
v_psse = psse_to_df('v_psse.xlsx')
omega_psse.iloc[:, 1:] += 1
omega_psse.iloc[:, 1:] *= 60

fig, ax = plot_comparison(ss2, ss2.GENROU.omega, omega,
                          a=(1,), a_tsat=[1], a_psse=[1],
                          ylabel="Generator Speed [Hz]",
                          title="WECC Scenario 2 (Line 4-16 Trip)",
```

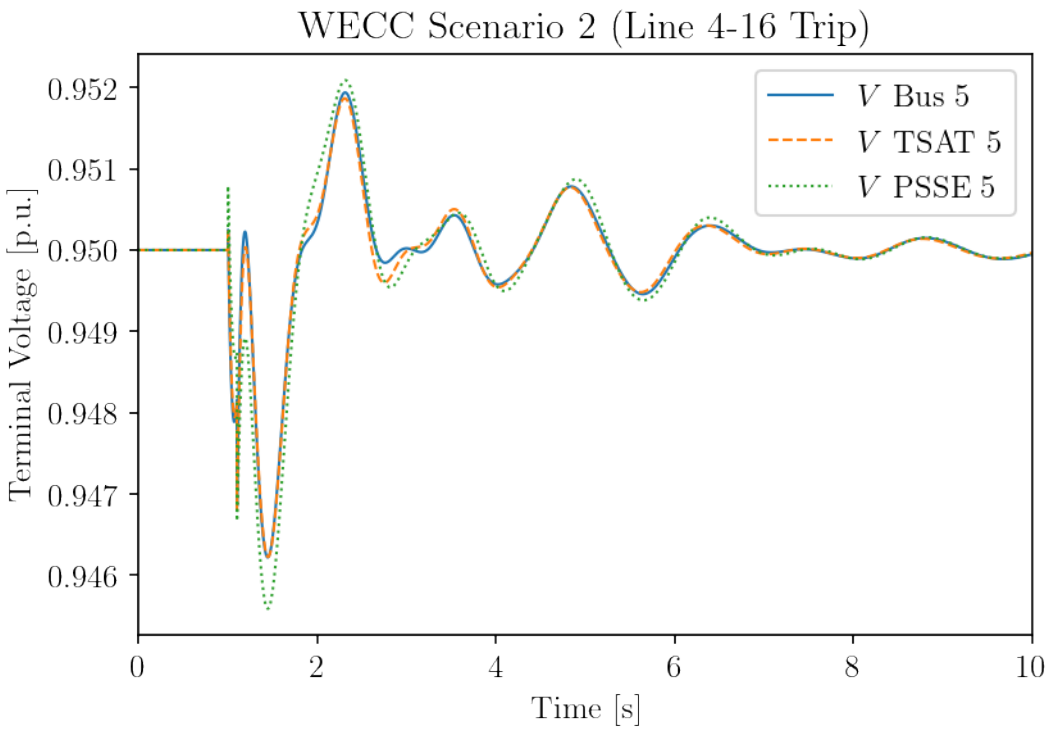
(continues on next page)

(continued from previous page)

```
                                tsat_header=tsat_omega_headers,
                                psse_data = omega_psse,
                                psse_header = psse_omega_headers,
                                scale=60, right=10,
                                legend=True,
                                show=False
                               )
```




```
fig, ax = plot_comparison(ss2, ss2.GENROU.v, v,
                                a=(1,), a_tsat=[1], a_psse=[1],
                                ylabel="Terminal Voltage [p.u.]",
                                title="WECC Scenario 2 (Line 4-16 Trip)",
                                tsat_header=tsat_v_headers,
                                psse_data = v_psse,
                                psse_header = psse_v_headers,
                                scale=1, right=10,
                                legend=True,
                                show=False
                               )
```



Appendix: CURENT WECC 179-Bus System Data

Power Flow Data (PSS/E RAW format)

```
data = !cat wecc.raw
print('\n'.join(data))
```

```
0,   100.00,  32, 0, 1, 60.00     / PSS(R)E 32 RAW created by rawd32  THU,␣
↪MAY 21 2020  10:24
WESTERN ELECTRICITY COORDINATING COUNCIL 179-BUS SYSTEM (MOD
DISTRIBUTED WITH ANDES (HTTPS://GITHUB.COM/CUIHANTAO/ANDES)
     1,'CORONADO    ', 500.0000,1,   1,   1,   1,0.97947, -26.1745
     2,'CHOLLA      ', 345.0000,1,   1,   1,   1,0.97744, -16.9603
     3,'CORONADO    ',  20.0000,2,   1,   1,   1,1.04000, -19.6589
     4,'CRAIG       ', 345.0000,1,   1,   1,   1,0.97518,  16.2755
     5,'CRAIG       ',  22.0000,2,   1,   1,   1,0.95000,  23.5536
     6,'FOURCORN    ', 500.0000,1,   1,   1,   1,1.06814,  -7.9038
     7,'FOURCORN    ', 345.0000,1,   1,   1,   1,1.00914,  -4.6874
     8,'FCNGN4CC    ',  22.0000,2,   1,   1,   1,1.00000,   2.2301
     9,'FOURCORN    ', 230.0000,1,   1,   1,   1,1.00726,  -5.2308
    10,'HAYDEN      ',  20.0000,2,   1,   1,   1,1.00000,  33.7299
    11,'NAVAJO      ', 500.0000,1,   2,   1,   1,1.07205, -23.9351
    12,'NAVAJO 2    ',  26.0000,2,   2,   1,   1,1.00000, -17.8112
    13,'MOENKOPI    ', 500.0000,1,   2,   1,   1,1.06735, -24.8548
    14,'PALOVRD2    ',  24.0000,2,   2,   1,   1,0.96000, -21.7139
    15,'PALOVRDE    ', 500.0000,1,   2,   1,   1,1.04856, -29.6406
```

(continues on next page)

(continued from previous page)

```
    16,'SAN JUAN    ', 345.0000,1,   1,   1,   1,1.03559,  -3.8744
    17,'SJUAN G4    ',  22.0000,2,   1,   1,   1,1.00000,  -0.8871
    18,'WESTWING    ', 500.0000,1,   2,   1,   1,1.05593, -29.5910
    19,'NAVAJO1     ', 500.0000,1,   2,   1,   1,1.07025, -22.8500
    20,'NAVAJO2     ', 500.0000,1,   2,   1,   1,1.06042, -25.9522
    21,'NAVAJO3     ', 500.0000,1,   2,   1,   1,1.04693, -22.1208
    22,'NAVAJO4     ', 500.0000,1,   2,   1,   1,1.02564, -31.4586
    23,'MOENKOP1    ', 500.0000,1,   2,   1,   1,1.05571, -23.3906
    24,'MOENKOP2    ', 500.0000,1,   2,   1,   1,1.04116, -31.0832
    25,'MOENKOP3    ', 500.0000,1,   2,   1,   1,1.05132, -15.5499
    26,'MOENKOP4    ', 500.0000,1,   2,   1,   1,1.04671, -42.3967
    27,'FOURCOR1    ', 500.0000,1,   1,   1,   1,1.06585,  -5.7792
    28,'FOURCOR2    ', 500.0000,1,   1,   1,   1,1.06933, -27.9822
    29,'CANAD G1    ',  20.0000,2,   1,   1,   1,1.00000,  24.7380
    30,'CANADA      ', 500.0000,1,   1,   1,   1,1.03634,  20.9474
    31,'CANALB      ', 500.0000,1,   1,   1,   1,1.07861,  49.2370
    32,'CA230TO     ', 230.0000,1,   1,   1,   1,0.97855,  53.8260
    33,'CA230       ', 230.0000,1,   1,   1,   1,1.00114,  62.8749
    34,'CMAIN GM    ',  20.0000,2,   1,   1,   1,1.02000,  67.7961
    35,'BRIDGER2    ',  22.0000,2,   1,   1,   1,1.00900,   2.4615
    36,'ADELANTO    ', 500.0000,1,   2,   1,   1,1.06032, -42.1554
    37,'ADELAN&1    ', 500.0000,1,   2,   1,   1,1.08090, -46.2256
    38,'CASTAIC     ', 230.0000,1,   2,   1,   1,1.03195, -47.2901
    39,'CASTAI4G    ',  18.0000,2,   2,   1,   1,1.02000, -45.9803
    40,'GLENDAL     ', 230.0000,1,   2,   1,   1,1.02337, -50.9223
    41,'HAYNES      ', 230.0000,1,   2,   1,   1,1.03129, -50.6520
    42,'HAYNES3G    ',  18.0000,2,   2,   1,   1,1.00000, -47.3488
    43,'INTERMT     ', 345.0000,1,   1,   1,   1,1.05257,  -4.3661
    44,'INTERM1G    ',  26.0000,2,   1,   1,   1,1.05000,   0.2811
    45,'OLIVE       ', 230.0000,1,   2,   1,   1,1.03475, -47.6289
    46,'OWENS G     ',  11.5000,2,   2,   1,   1,1.02000, -47.2682
    47,'RINALDI     ', 230.0000,1,   2,   1,   1,1.03402, -47.8387
    48,'RINALDI     ', 500.0000,1,   2,   1,   1,1.07212, -45.4811
    49,'RIVER       ', 230.0000,1,   2,   1,   1,1.02362, -51.8438
    50,'STA B       ', 138.0000,1,   2,   1,   1,1.03256, -51.9003
    51,'STA B1      ', 287.0000,1,   2,   1,   1,1.03663, -50.7167
    52,'STA B2      ', 287.0000,1,   2,   1,   1,1.03663, -50.7167
    53,'STA BLD     ', 230.0000,1,   2,   1,   1,1.02728, -52.1086
    54,'STA E       ', 230.0000,1,   2,   1,   1,1.02516, -50.0567
    55,'STA E       ', 500.0000,1,   2,   1,   1,1.04304, -47.6632
    56,'STA F       ', 230.0000,1,   2,   1,   1,1.02484, -51.8789
    57,'STA G       ', 230.0000,1,   2,   1,   1,1.02394, -51.1740
    58,'STA J       ', 230.0000,1,   2,   1,   1,1.03112, -48.9133
    59,'SYLMARLA    ', 230.0000,1,   2,   1,   1,1.03832, -47.1488
    60,'SYLMAR S    ', 230.0000,1,   2,   1,   1,1.02073, -48.2552
    61,'VALLEY      ', 230.0000,1,   2,   1,   1,1.02930, -49.0304
    62,'VICTORVL    ', 287.0000,1,   2,   1,   1,1.05209, -44.4925
    63,'VICTORVL    ', 500.0000,1,   2,   1,   1,1.05866, -42.4751
    64,'MONTA G1    ',  20.0000,2,   1,   1,   1,1.00000,  56.5947
    65,'MONTANA     ', 500.0000,1,   1,   1,   1,1.04928,  48.2022
    66,'BIG EDDY    ', 115.0000,1,   1,   1,   1,1.06863, -16.9326
    67,'BIG EDDY    ', 230.0000,1,   1,   1,   1,1.06592, -14.5699
```

(continues on next page)

(continued from previous page)

```
  68,'BIG EDDY    ', 500.0000,1,   1,   1,   1,1.08919, -13.3098
  69,'DALLES21    ',  13.8000,2,   1,   1,   1,1.05500,  -7.9400
  70,'CELILO      ', 230.0000,1,   1,   1,   1,1.06159, -15.0760
  71,'CELILOCA    ', 500.0000,1,   1,   1,   1,1.08964, -13.4178
  72,'COLSTRP     ', 500.0000,1,   1,   1,   1,1.07839,  -1.3595
  73,'COULEE      ', 500.0000,1,   1,   1,   1,1.07000,   0.1645
  74,'GARRISON    ', 500.0000,1,   1,   1,   1,1.03705, -12.1586
  75,'JOHN DAY    ', 500.0000,1,   1,   1,   1,1.08283, -11.1230
  76,'JOHN DAY    ',  13.8000,3,   1,   1,   1,1.00000,   0.0000
  77,'HANFORD     ', 500.0000,1,   1,   1,   1,1.04947,   0.2258
  78,'NORTH G3    ',  20.0000,2,   1,   1,   1,1.00000,  26.5872
  79,'NORTH       ', 500.0000,1,   1,   1,   1,1.04993,  12.1085
  80,'BURNS       ', 500.0000,1,   1,   1,   1,1.05295, -19.5513
  81,'GRIZZLY     ', 500.0000,1,   1,   1,   1,1.06740, -17.1477
  82,'MALIN       ', 500.0000,1,   1,   1,   1,1.05433, -23.3455
  83,'MIDPOINT    ', 500.0000,1,   1,   1,   1,1.06195,  -5.1981
  84,'MIDPOINT    ', 345.0000,1,   1,   1,   1,0.99845,  -1.5707
  85,'SUMMER L    ', 500.0000,1,   1,   1,   1,1.05818, -18.2347
  86,'GRIZZLY1    ', 500.0000,1,   1,   1,   1,1.06533, -18.7474
  87,'GRIZZLY2    ', 500.0000,1,   1,   1,   1,1.07143, -16.0156
  88,'MALIN1      ', 500.0000,1,   1,   1,   1,1.05062, -14.1380
  89,'MALIN2      ', 500.0000,1,   1,   1,   1,1.06088, -19.8058
  90,'GRIZZLY3    ', 500.0000,1,   1,   1,   1,1.07997, -21.7956
  91,'GRIZZLY4    ', 500.0000,1,   1,   1,   1,1.06943, -17.2249
  92,'GRIZZLY5    ', 500.0000,1,   1,   1,   1,1.06860, -22.4488
  93,'GRIZZLY6    ', 500.0000,1,   1,   1,   1,1.07170, -17.8609
  94,'GRIZZLY7    ', 500.0000,1,   1,   1,   1,1.07926, -22.0559
  95,'GRIZZLY8    ', 500.0000,1,   1,   1,   1,1.06917, -17.2247
  96,'GRIZZLY9    ', 500.0000,1,   1,   1,   1,1.06761, -22.7454
  97,'GRIZZLYA    ', 500.0000,1,   1,   1,   1,1.07104, -17.8942
  98,'BURNS1      ', 500.0000,1,   1,   1,   1,0.97284,   8.0442
  99,'CORTINA     ', 200.0000,1,   3,   1,   1,1.13097, -34.9741
 100,'COTWDPGE    ', 200.0000,1,   3,   1,   1,1.13613, -30.4883
 101,'DIABLO      ', 500.0000,1,   2,   1,   1,1.05298, -46.1260
 102,'DIABLO1     ',  25.0000,2,   2,   1,   1,0.98000, -42.3238
 103,'GATES       ', 500.0000,1,   2,   1,   1,1.04707, -47.6820
 104,'GLENN       ', 200.0000,1,   3,   1,   1,1.14090, -34.1986
 105,'LOGAN CR    ', 200.0000,1,   3,   1,   1,1.14016, -34.5836
 106,'LOSBANOS    ', 500.0000,1,   2,   1,   1,1.04927, -49.5273
 107,'MIDWAY      ', 500.0000,1,   2,   1,   1,1.05934, -48.6051
 108,'MIDWAY      ', 200.0000,1,   2,   1,   1,1.16705, -51.4430
 109,'MOSSLAND    ', 500.0000,1,   2,   1,   1,1.04639, -49.7921
 110,'OLINDA      ', 500.0000,1,   3,   1,   1,1.03727, -31.0686
 111,'ROUND MT    ',  20.0000,2,   3,   1,   1,1.02000, -15.0791
 112,'ROUND MT    ', 200.0000,1,   3,   1,   1,1.12385, -25.1402
 113,'ROUND MT    ', 500.0000,1,   3,   1,   1,1.03454, -27.9316
 114,'TABLE MT    ', 500.0000,1,   3,   1,   1,1.01342, -32.0699
 115,'TEVATR      ',  20.0000,2,   3,   1,   1,1.05000, -35.6850
 116,'TEVATR      ', 200.0000,1,   3,   1,   1,1.12744, -39.6332
 117,'TEVATR2     ',  20.0000,2,   3,   1,   1,1.00000, -30.7816
 118,'TEVATR      ', 500.0000,1,   3,   1,   1,0.99816, -38.9938
 119,'OLINDA1     ', 500.0000,1,   3,   1,   1,1.05026, -24.0934
```

(continues on next page)

(continued from previous page)

```
 120,'OLINDA2     ', 500.0000,1,   3,   1,   1,1.01142, -38.0294
 121,'OLINDA3     ', 500.0000,1,   3,   1,   1,1.02786, -30.7891
 122,'OLINDA4     ', 500.0000,1,   3,   1,   1,0.98098, -46.5990
 123,'ROUND1      ', 500.0000,1,   3,   1,   1,1.04054, -22.0273
 124,'ROUND2      ', 500.0000,1,   3,   1,   1,1.01079, -36.1445
 125,'ROUND3      ', 500.0000,1,   3,   1,   1,1.04054, -22.0273
 126,'ROUND4      ', 500.0000,1,   3,   1,   1,1.01079, -36.1445
 127,'TABLE1      ', 500.0000,1,   3,   1,   1,1.01257, -25.8067
 128,'TABLE2      ', 500.0000,1,   3,   1,   1,0.99882, -45.3405
 129,'TABLE3      ', 500.0000,1,   3,   1,   1,1.01755, -27.3173
 130,'TABLE4      ', 500.0000,1,   3,   1,   1,0.99876, -41.4218
 131,'TEVATR1     ', 500.0000,1,   2,   1,   1,0.97999, -36.9419
 132,'TEVATR2     ', 500.0000,1,   2,   1,   1,1.07047, -51.7010
 133,'TEVATR3     ', 500.0000,1,   2,   1,   1,0.97141, -35.0590
 134,'GATES1      ', 500.0000,1,   2,   1,   1,1.06807, -49.9922
 135,'DEVERS      ', 500.0000,1,   2,   1,   1,1.03539, -43.5212
 136,'EAGLROCK    ', 230.0000,1,   2,   1,   1,1.01010, -52.7598
 137,'ELDORADO    ',  20.0000,2,   2,   1,   1,1.02000, -25.9732
 138,'ELDORADO    ', 500.0000,1,   2,   1,   1,1.05122, -33.0947
 139,'LITEHIPE    ',  20.0000,2,   2,   1,   1,1.02000, -49.6246
 140,'LITEHIPE    ', 230.0000,1,   2,   1,   1,1.01186, -55.7741
 141,'LUGO        ', 500.0000,1,   2,   1,   1,1.05498, -46.0893
 142,'MESA CAL    ', 230.0000,1,   2,   1,   1,1.00705, -55.0594
 143,'MIRALOMA    ',  20.0000,2,   2,   1,   1,1.05000, -45.4414
 144,'MIRALOMA    ', 500.0000,1,   2,   1,   1,1.04087, -49.6789
 145,'MIRALOMA    ', 230.0000,1,   2,   1,   1,1.03821, -50.1247
 146,'MOHAVE      ', 500.0000,1,   2,   1,   1,1.06999, -29.3603
 147,'MOHAV1CC    ',  22.0000,2,   2,   1,   1,1.05000, -21.0397
 148,'PARDEE      ',  20.0000,2,   2,   1,   1,1.01000, -39.5994
 149,'PARDEE      ', 230.0000,1,   2,   1,   1,1.00600, -51.6818
 150,'SERRANO     ', 500.0000,1,   2,   1,   1,1.04129, -50.0775
 151,'VALLEY      ', 500.0000,1,   2,   1,   1,1.03694, -47.7607
 152,'VINCENT     ', 500.0000,1,   2,   1,   1,1.06119, -48.9450
 153,'VINCENT     ', 230.0000,1,   2,   1,   1,0.99489, -51.6324
 154,'CAMP WIL    ', 345.0000,1,   1,   1,   1,1.04289,  -2.4160
 155,'BENLOMND    ', 345.0000,1,   1,   1,   1,1.04464,  -3.3929
 156,'BENLOMND    ', 230.0000,1,   1,   1,   1,1.04624,  -3.9585
 157,'EMERY       ', 345.0000,1,   1,   1,   1,1.03705,   4.8898
 158,'EMERY       ',  20.0000,2,   1,   1,   1,1.05000,   9.6052
 159,'MONA        ', 345.0000,1,   1,   1,   1,1.05596,  -2.0072
 160,'NAUGHTON    ', 230.0000,1,   1,   1,   1,1.04453,   0.6143
 161,'NAUGHT      ',  20.0000,2,   1,   1,   1,1.00000,   3.4140
 162,'PINTO PS    ', 345.0000,1,   1,   1,   1,1.03679,  -2.4413
 163,'PINTO       ', 345.0000,1,   1,   1,   1,1.04044,  -1.4027
 164,'SPAN FRK    ', 345.0000,1,   1,   1,   1,1.03514,  -1.0920
 165,'SIGURD      ', 345.0000,1,   1,   1,   1,1.05190,  -0.5455
 166,'TERMINAL    ', 345.0000,1,   1,   1,   1,1.03911,  -3.4207
 167,'MALIN3      ', 500.0000,1,   3,   1,   1,1.06106, -19.7689
 168,'MALIN4      ', 500.0000,1,   3,   1,   1,1.02557, -31.6678
 169,'MALIN5      ', 500.0000,1,   3,   1,   1,1.06179, -19.0138
 170,'MALIN6      ', 500.0000,1,   3,   1,   1,1.02702, -31.6971
 171,'MALIN7      ', 500.0000,1,   3,   1,   1,1.05696, -17.4396
```

(continues on next page)

(continued from previous page)

```
  172,'MALIN8      ', 500.0000,1,   3,   1,   1,1.03194, -37.1499
  173,'MIDWAY1     ', 500.0000,1,   2,   1,   1,1.04651, -48.2385
  174,'MIDWAY2     ', 500.0000,1,   2,   1,   1,1.05458, -49.3078
  175,'MIDWAY3     ', 500.0000,1,   2,   1,   1,1.04625, -48.2334
  176,'MIDWAY4     ', 500.0000,1,   2,   1,   1,1.05470, -49.3092
  177,'MIDWAY5     ', 500.0000,1,   2,   1,   1,1.04598, -48.2211
  178,'MIDWAY6     ', 500.0000,1,   2,   1,   1,1.05705, -49.2854
  179,'BURNS2      ', 500.0000,1,   1,   1,   1,0.98437,  -6.6858
0 /End of Bus data, Begin Load data
    1,'BL',1,   1,   1,  1750.000,   -56.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
    3,'BL',1,   1,   1,   100.000,     0.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
    4,'BL',1,   1,   1,  2350.000,  -127.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
    5,'BL',1,   1,   1,   100.000,     0.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
    7,'BL',1,   1,   1,   239.000,   -56.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
    8,'BL',1,   1,   1,   100.000,     0.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
    9,'BL',1,   1,   1,   139.700,    23.800,     0.000,     0.000,     0.
↪000,     0.000,   1,1
   10,'BL',1,   1,   1,   100.000,     0.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
   11,'BL',1,   2,   1,    90.000,    70.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
   12,'BL',1,   2,   1,   100.000,     0.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
   14,'BL',1,   2,   1,   100.000,     0.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
   15,'BL',1,   2,   1,   793.400,   207.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
   16,'BL',1,   1,   1,   840.000,     5.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
   17,'BL',1,   1,   1,   100.000,     0.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
   18,'BL',1,   2,   1,   617.000,   -69.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
   29,'BL',1,   1,   1,   100.000,     0.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
   30,'BL',1,   1,   1,  4400.000,  1000.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
   33,'BL',1,   1,   1,  3600.000,   700.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
   34,'BL',1,   1,   1,   100.000,     0.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
   35,'BL',1,   1,   1,   100.000,     0.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
   36,'BL',1,   2,   1, -1862.000,   971.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
   39,'BL',1,   2,   1,   100.000,     0.000,     0.000,     0.000,     0.
```

(continues on next page)

(continued from previous page)

```
↪000,      0.000,   1,1
    40,'BL',1,   2,   1,   135.000,    27.000,     0.000,     0.000,     0.
↪000,      0.000,   1,1
    42,'BL',1,   2,   1,   100.000,     0.000,     0.000,     0.000,     0.
↪000,      0.000,   1,1
    43,'BL',1,   1,   1,  2053.000,   907.100,     0.000,     0.000,     0.
↪000,      0.000,   1,1
    44,'BL',1,   1,   1,   100.000,     0.000,     0.000,     0.000,     0.
↪000,      0.000,   1,1
    45,'BL',1,   2,   1,   -72.800,   -17.000,     0.000,     0.000,     0.
↪000,      0.000,   1,1
    46,'BL',1,   2,   1,   100.000,     0.000,     0.000,     0.000,     0.
↪000,      0.000,   1,1
    47,'BL',1,   2,   1,   121.000,    25.000,     0.000,     0.000,     0.
↪000,      0.000,   1,1
    49,'BL',1,   2,   1,   320.000,    65.000,     0.000,     0.000,     0.
↪000,      0.000,   1,1
    50,'BL',1,   2,   1,   237.200,   -63.200,     0.000,     0.000,     0.
↪000,      0.000,   1,1
    53,'BL',1,   2,   1,   138.000,    28.000,     0.000,     0.000,     0.
↪000,      0.000,   1,1
    54,'BL',1,   2,   1,   807.800,   132.100,     0.000,     0.000,     0.
↪000,      0.000,   1,1
    56,'BL',1,   2,   1,   117.000,    24.000,     0.000,     0.000,     0.
↪000,      0.000,   1,1
    57,'BL',1,   2,   1,   121.000,    25.000,     0.000,     0.000,     0.
↪000,      0.000,   1,1
    58,'BL',1,   2,   1,   887.700,    -6.200,     0.000,     0.000,     0.
↪000,      0.000,   1,1
    59,'BL',1,   2,   1, -2771.000,  1654.000,     0.000,     0.000,     0.
↪000,      0.000,   1,1
    60,'BL',1,   2,   1,   401.000,    80.600,     0.000,     0.000,     0.
↪000,      0.000,   1,1
    61,'BL',1,   2,   1,   205.200,    17.600,     0.000,     0.000,     0.
↪000,      0.000,   1,1
    62,'BL',1,   2,   1,  -129.000,    32.200,     0.000,     0.000,     0.
↪000,      0.000,   1,1
    64,'BL',1,   1,   1,   100.000,     0.000,     0.000,     0.000,     0.
↪000,      0.000,   1,1
    65,'BL',1,   1,   1,  1700.000,   300.000,     0.000,     0.000,     0.
↪000,      0.000,   1,1
    66,'BL',1,   1,   1,   160.000,    31.250,     0.000,     0.000,     0.
↪000,      0.000,   1,1
    67,'BL',1,   1,   1,   -67.500,   160.000,     0.000,     0.000,     0.
↪000,      0.000,   1,1
    68,'BL',1,   1,   1,   -44.200,    22.000,     0.000,     0.000,     0.
↪000,      0.000,   1,1
    69,'BL',1,   1,   1,   100.000,     0.000,     0.000,     0.000,     0.
↪000,      0.000,   1,1
    70,'BL',1,   1,   1,  3137.000,  1681.000,     0.000,     0.000,     0.
↪000,      0.000,   1,1
    72,'BL',1,   1,   1, -1525.000,   -50.000,     0.000,     0.000,     0.
```

(continues on next page)

(continued from previous page)

```
↪000,     0.000,   1,1
    74,'BL',1,   1,   1,  2584.000,   394.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
    75,'BL',1,   1,   1,  3200.000,  1100.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
    76,'BL',1,   1,   1,   100.000,     0.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
    77,'BL',1,   1,   1,  3500.000,   500.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
    78,'BL',1,   1,   1,   100.000,     0.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
    79,'BL',1,   1,   1,  5000.000,   400.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
    81,'BL',1,   1,   1,   -66.600,   -97.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
    82,'BL',1,   1,   1,  -339.000,  -119.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
    84,'BL',1,   1,   1,   610.000,  -414.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
    99,'BL',1,   3,   1,   -43.300,    20.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
   100,'BL',1,   3,   1,   210.400,   -77.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
   101,'BL',1,   2,   1,    50.000,    25.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
   102,'BL',1,   2,   1,   100.000,     0.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
   103,'BL',1,   2,   1,   305.000,    -7.600,     0.000,     0.000,     0.
↪000,     0.000,   1,1
   104,'BL',1,   3,   1,    27.500,    -0.100,     0.000,     0.000,     0.
↪000,     0.000,   1,1
   105,'BL',1,   3,   1,     8.010,     0.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
   106,'BL',1,   2,   1,   265.000,    14.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
   107,'BL',1,   2,   1,    55.600,  -329.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
   108,'BL',1,   2,   1,   777.600,    32.600,     0.000,     0.000,     0.
↪000,     0.000,   1,1
   109,'BL',1,   2,   1,    40.000,    21.500,     0.000,     0.000,     0.
↪000,     0.000,   1,1
   110,'BL',1,   3,   1,  -189.000,    61.500,     0.000,     0.000,     0.
↪000,     0.000,   1,1
   111,'BL',1,   3,   1,   100.000,     0.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
   112,'BL',1,   3,   1,   148.000,     0.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
   114,'BL',1,   3,   1,    -0.700,   118.500,     0.000,     0.000,     0.
↪000,     0.000,   1,1
   115,'BL',1,   3,   1,   100.000,     0.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
   116,'BL',1,   3,   1,   884.000,    54.800,     0.000,     0.000,     0.
```

(continues on next page)

(continued from previous page)

```
↪000,     0.000,   1,1
  117,'BL',1,   3,   1,   100.000,     0.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
  118,'BL',1,   3,   1,  5661.000,  3491.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
  135,'BL',1,   2,   1,   856.000,    19.600,     0.000,     0.000,     0.
↪000,     0.000,   1,1
  136,'BL',1,   2,   1,   175.000,    18.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
  137,'BL',1,   2,   1,   100.000,     0.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
  138,'BL',1,   2,   1,   902.300,   -11.400,     0.000,     0.000,     0.
↪000,     0.000,   1,1
  139,'BL',1,   2,   1,   100.000,     0.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
  140,'BL',1,   2,   1,  3191.000,   630.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
  141,'BL',1,   2,   1,   204.200,   -28.200,     0.000,     0.000,     0.
↪000,     0.000,   1,1
  142,'BL',1,   2,   1,   377.400,    64.500,     0.000,     0.000,     0.
↪000,     0.000,   1,1
  143,'BL',1,   2,   1,   100.000,     0.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
  144,'BL',1,   2,   1,  3098.000,  1189.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
  147,'BL',1,   2,   1,   100.000,     0.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
  148,'BL',1,   2,   1,   100.000,     0.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
  149,'BL',1,   2,   1,  3118.000,    78.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
  150,'BL',1,   2,   1,  1230.000,    72.800,     0.000,     0.000,     0.
↪000,     0.000,   1,1
  151,'BL',1,   2,   1,   406.000,    41.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
  153,'BL',1,   2,   1,  1066.000,   -10.800,     0.000,     0.000,     0.
↪000,     0.000,   1,1
  154,'BL',1,   1,   1,   457.700,    81.700,     0.000,     0.000,     0.
↪000,     0.000,   1,1
  155,'BL',1,   1,   1,    33.900,    11.900,     0.000,     0.000,     0.
↪000,     0.000,   1,1
  156,'BL',1,   1,   1,   148.000,    -7.900,     0.000,     0.000,     0.
↪000,     0.000,   1,1
  157,'BL',1,   1,   1,   116.100,    38.400,     0.000,     0.000,     0.
↪000,     0.000,   1,1
  158,'BL',1,   1,   1,   100.000,     0.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
  159,'BL',1,   1,   1,   -62.000,    12.800,     0.000,     0.000,     0.
↪000,     0.000,   1,1
  160,'BL',1,   1,   1,   255.000,   100.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
  161,'BL',1,   1,   1,   100.000,     0.000,     0.000,     0.000,     0.
```

(continues on next page)

(continued from previous page)

```
↪000,     0.000,   1,1
  163,'BL',1,   1,   1,    31.600,    11.500,     0.000,     0.000,     0.
↪000,     0.000,   1,1
  164,'BL',1,   1,   1,   141.200,    71.400,     0.000,     0.000,     0.
↪000,     0.000,   1,1
  165,'BL',1,   1,   1,   379.000,   -43.000,     0.000,     0.000,     0.
↪000,     0.000,   1,1
  166,'BL',1,   1,   1,   185.000,    78.500,     0.000,     0.000,     0.
↪000,     0.000,   1,1
0 /End of Load data, Begin Fixed shunt data
    6,'1 ',1,     0.000,  -113.000
    7,'1 ',1,     0.000,  -155.000
   11,'1 ',1,     0.000,  -190.000
   13,'1 ',1,     0.000,  -391.000
   15,'1 ',1,     0.000,  -146.000
   16,'1 ',1,     0.000,   390.000
   18,'1 ',1,     0.000,  -427.000
   36,'1 ',1,     0.000,   912.000
   43,'1 ',1,     0.000,   430.000
   48,'1 ',1,     0.000,   -80.000
   59,'1 ',1,     0.000,  2146.000
   62,'1 ',1,     0.000,  -108.000
   67,'1 ',1,     0.000,   576.850
   70,'1 ',1,     0.000,   792.000
   71,'1 ',1,     0.000,   462.000
   75,'1 ',1,     0.000,  1019.350
   77,'1 ',1,     0.000,   550.000
   79,'1 ',1,     0.000,  1200.000
   80,'1 ',1,     0.000,  -220.000
   81,'1 ',1,     0.000,  -674.000
   82,'1 ',1,     0.000,  -110.000
   83,'1 ',1,     0.000,  -220.000
   84,'1 ',1,     0.000,  -870.000
  103,'1 ',1,     0.000,   -91.000
  107,'1 ',1,     0.000,  -327.000
  108,'1 ',1,     0.000,  -130.000
  112,'1 ',1,     0.000,  -128.000
  113,'1 ',1,     0.000,   -91.000
  114,'1 ',1,     0.000,   -91.000
  116,'1 ',1,     0.000,   -32.000
  118,'1 ',1,     0.000,  1500.000
  138,'1 ',1,     0.000,  -319.000
  144,'1 ',1,     0.000,   400.000
  146,'1 ',1,     0.000,  -196.000
  153,'1 ',1,     0.000,  -190.000
  154,'1 ',1,     0.000,   -60.000
  157,'1 ',1,     0.000,  -220.000
  163,'1 ',1,     0.000,   -18.000
  165,'1 ',1,     0.000,   -50.000
  179,'1 ',1,     0.000,  -220.000
0 /End of Fixed shunt data, Begin Generator data
    3,'1 ',   800.000,   123.043,   300.000,  -300.000,1.04000,     0,  1600.
```

(continues on next page)

(continued from previous page)

```
↪000, 0.00000E+0, 2.50000E-1, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000,     0.000,   1,1.0000
    5,'1 ',  1048.000,  -132.905,   400.000,  -400.000,0.95000,     0,  2100.
↪000, 0.00000E+0, 2.50000E-1, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000,     0.000,   1,1.0000
    8,'1 ',  2160.000,   -30.469,   700.000,  -500.000,1.00000,     0,  4300.
↪000, 0.00000E+0, 2.50000E-1, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000,     0.000,   1,1.0000
   10,'1 ',  2050.000,   464.829,   900.000,  -900.000,1.00000,     0,  4100.
↪000, 0.00000E+0, 2.50000E-1, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000,     0.000,   1,1.0000
   12,'1 ',  1690.000,   195.571,   700.000,  -280.000,1.00000,     0,  3400.
↪000, 0.00000E+0, 2.50000E-1, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000,     0.000,   1,1.0000
   14,'1 ',  2640.000,   378.080,  1300.000,  -900.000,0.96000,     0,  5300.
↪000, 0.00000E+0, 2.50000E-1, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000,     0.000,   1,1.0000
   17,'1 ',   962.000,   148.778,   300.000,  -300.000,1.00000,     0,  2000.
↪000, 0.00000E+0, 2.50000E-1, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000,     0.000,   1,1.0000
   29,'1 ',  4450.000,  1011.075,  4000.000, -4000.000,1.00000,     0,  8900.
↪000, 0.00000E+0, 2.50000E-1, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000,     0.000,   1,1.0000
   34,'1 ',  4480.000,  1150.177,  5320.000, -3500.000,1.02000,     0,  9000.
↪000, 0.00000E+0, 2.50000E-1, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000,     0.000,   1,1.0000
   35,'1 ',  1640.000,   285.666,   600.000,  -525.000,1.00900,     0,  3300.
↪000, 0.00000E+0, 2.50000E-1, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000,     0.000,   1,1.0000
   39,'1 ',   200.000,   -52.159,   268.000,  -134.000,1.02000,     0,   400.
↪000, 0.00000E+0, 2.50000E-1, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000,     0.000,   1,1.0000
   42,'1 ',   325.000,    68.266,   300.000,  -220.000,1.00000,     0,   650.
↪000, 0.00000E+0, 2.50000E-1, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000,     0.000,   1,1.0000
   44,'1 ',  1780.000,   534.593,   850.000,  -440.000,1.05000,     0,  3600.
↪000, 0.00000E+0, 2.50000E-1, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000,     0.000,   1,1.0000
   46,'1 ',   110.000,    29.078,   100.000,  -100.000,1.02000,     0,   220.
↪000, 0.00000E+0, 2.50000E-1, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000,     0.000,   1,1.0000
   64,'1 ',  2910.000,   953.299,  1500.000, -1000.000,1.00000,     0,  5800.
↪000, 0.00000E+0, 2.50000E-1, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000,     0.000,   1,1.0000
   69,'1 ',  1301.000,   431.493,   692.000,  -711.000,1.05500,     0,  2600.
↪000, 0.00000E+0, 2.50000E-1, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000,     0.000,   1,1.0000
   76,'1 ',  5174.765,   855.276,  2649.000, -1850.000,1.00000,     0, 10400.
↪000, 0.00000E+0, 2.50000E-1, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000,     0.000,   1,1.0000
   78,'1 ',  9950.000,  1854.073,  5780.000, -2000.000,1.00000,     0, 20000.
↪000, 0.00000E+0, 2.50000E-1, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
```

(continues on next page)

(continued from previous page)

```
↪9999.000,     0.000,   1,1.0000
  102,'1 ',   765.000,  -206.264,   330.000,  -310.000,0.98000,     0,  1500.
↪000, 0.00000E+0, 2.50000E-1, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000,     0.000,   1,1.0000
  111,'1 ',  1057.000,    25.675,   400.000,  -400.000,1.02000,     0,  2100.
↪000, 0.00000E+0, 2.50000E-1, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000,     0.000,   1,1.0000
  115,'1 ',   594.000,   192.348,   300.000,  -300.000,1.05000,     0,  1200.
↪000, 0.00000E+0, 2.50000E-1, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000,     0.000,   1,1.0000
  117,'1 ',  3467.000,  1654.420,  2500.000, -1000.000,1.00000,     0,  6900.
↪000, 0.00000E+0, 2.50000E-1, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000,     0.000,   1,1.0000
  137,'1 ',   982.700,  -128.764,   300.000,  -300.000,1.02000,     0,  2000.
↪000, 0.00000E+0, 2.50000E-1, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000,     0.000,   1,1.0000
  139,'1 ',  3195.000,  1032.512,  2000.000,  -900.000,1.02000,     0,  6400.
↪000, 0.00000E+0, 2.50000E-1, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000,     0.000,   1,1.0000
  143,'1 ',  1690.000,   593.821,   900.000,  -400.000,1.05000,     0,  3400.
↪000, 0.00000E+0, 2.50000E-1, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000,     0.000,   1,1.0000
  147,'1 ',  1680.000,   446.627,   700.000,  -300.000,1.05000,     0,  3400.
↪000, 0.00000E+0, 2.50000E-1, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000,     0.000,   1,1.0000
  148,'1 ',  2200.000,   393.721,   600.000,  -600.000,1.01000,     0,  4400.
↪000, 0.00000E+0, 2.50000E-1, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000,     0.000,   1,1.0000
  158,'1 ',  1665.000,   -31.368,  9999.000, -9999.000,1.05000,     0,  3300.
↪000, 0.00000E+0, 2.50000E-1, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000,     0.000,   1,1.0000
  161,'1 ',   445.000,    91.733,  9999.000, -9999.000,1.00000,     0,   900.
↪000, 0.00000E+0, 2.50000E-1, 0.00000E+0, 0.00000E+0,1.00000,1,  100.0, ␣
↪9999.000,     0.000,   1,1.0000
 0 /End of Generator data, Begin Branch data
    2,      7,'1 ', 1.79000E-3, 1.98800E-2,   2.57600,    0.00,    0.00,   ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    4,     16,'1 ', 9.77000E-3, 1.10000E-1,   2.00000,    0.00,    0.00,   ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    4,    159,'1 ', 8.11000E-3, 1.36900E-1,   2.43480,    0.00,    0.00,   ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    6,     27,'1 ',-0.00000E+0,-4.08000E-3,   0.00000,    0.00,    0.00,   ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    7,     16,'1 ', 5.00000E-4, 5.30000E-3,   0.08820,    0.00,    0.00,   ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    7,    162,'1 ', 4.80000E-3, 4.36000E-2,   0.70780,    0.00,    0.00,   ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
   11,     19,'1 ',-0.00000E+0,-6.34000E-3,   0.00000,    0.00,    0.00,   ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
   11,     21,'1 ',-0.00000E+0,-1.18800E-2,   0.00000,    0.00,    0.00,   ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
   11,    138,'1 ', 2.80000E-3, 2.11000E-2,   1.01940,    0.00,    0.00,␣
```

(continues on next page)

(continued from previous page)

```
↪1630.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    13,     20,'1 ',-0.00000E+0,-6.34000E-3,   0.00000,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    13,     23,'1 ',-0.00000E+0,-8.26000E-3,   0.00000,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    13,     25,'1 ',-0.00000E+0,-1.79500E-2,   0.00000,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    13,     28,'1 ',-0.00000E+0,-6.12000E-3,   0.00000,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    15,     18,'1 ', 4.00000E-4, 9.60000E-3,   0.90380,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    15,     18,'2 ', 4.00000E-4, 9.60000E-3,   0.90380,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    15,    135,'1 ', 2.59000E-3, 2.96700E-2,   2.15300,    0.00,    0.00,␣
↪1800.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    15,    135,'2 ', 2.59000E-3, 2.96700E-2,   2.15300,    0.00,    0.00,␣
↪1800.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    18,     22,'1 ',-0.00000E+0,-1.18800E-2,   0.00000,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    18,     24,'1 ',-0.00000E+0,-8.26000E-3,   0.00000,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    19,     20,'1 ', 7.70000E-4, 1.80400E-2,   1.39842,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    21,     22,'1 ', 2.41000E-3, 5.86500E-2,   4.86560,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    23,     24,'1 ', 1.79000E-3, 4.24400E-2,   3.39220,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    25,     26,'1 ', 2.07000E-3, 4.95900E-2,   3.95160,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    26,    138,'1 ',-0.00000E+0,-1.79500E-2,   0.00000,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    27,     28,'1 ', 1.77000E-3, 4.18900E-2,   3.34460,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    30,     31,'1 ', 3.50000E-3, 7.00000E-2,   4.60600,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    30,     79,'1 ', 8.30000E-4, 2.39000E-2,   3.30000,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    32,     33,'1 ', 2.00000E-3, 2.00000E-2,   0.80000,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    36,     37,'1 ', 7.40000E-4, 1.86100E-2,   1.40264,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    36,     55,'1 ', 8.20000E-4, 1.66800E-2,   1.18802,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    36,     63,'1 ',-0.00000E+0, 1.59000E-3,   0.12002,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    36,     63,'2 ',-0.00000E+0, 1.59000E-3,   0.12002,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    38,     45,'1 ', 2.21000E-3, 3.34600E-2,   0.07338,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    38,     47,'1 ', 2.90000E-3, 3.80000E-2,   0.08240,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    38,     58,'1 ', 3.09000E-3, 4.67700E-2,   0.10080,    0.00,    0.00,  ␣
```

(continues on next page)

(continued from previous page)

```
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    38,     59,'1 ', 2.26000E-3, 3.42200E-2,   0.07506,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    40,     54,'1 ', 4.70000E-4, 7.23000E-3,   0.01624,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    40,     57,'1 ', 3.50000E-4, 5.36000E-3,   0.01204,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    41,     49,'1 ', 2.20000E-3, 3.42200E-2,   0.07716,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    41,     49,'2 ', 2.38000E-3, 3.66900E-2,   0.08284,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    41,     56,'1 ', 2.01000E-3, 3.07400E-2,   0.06886,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    41,     57,'1 ', 2.81000E-3, 4.29600E-2,   0.09648,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    43,    159,'1 ', 1.80000E-3, 2.45000E-2,   0.43920,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    43,    159,'2 ', 1.80000E-3, 2.45000E-2,   0.43920,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    45,     47,'1 ', 2.90000E-4, 4.34000E-3,   0.00950,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    47,     54,'1 ', 2.29000E-3, 1.58300E-2,   0.03060,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    47,     54,'2 ', 2.29000E-3, 1.58300E-2,   0.03060,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    47,     58,'1 ', 1.41000E-3, 9.67000E-3,   0.01940,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    47,     58,'2 ', 1.41000E-3, 9.67000E-3,   0.01940,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    47,     58,'3 ', 1.61000E-3, 9.71000E-3,   0.01928,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    47,     58,'4 ', 1.61000E-3, 9.71000E-3,   0.01928,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    47,     59,'1 ', 2.70000E-4, 3.93000E-3,   0.00918,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    47,     59,'2 ', 2.70000E-4, 3.93000E-3,   0.00918,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    47,     59,'3 ', 2.70000E-4, 3.93000E-3,   0.00918,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    47,     61,'1 ', 1.38000E-3, 1.11600E-2,   0.02470,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    47,     61,'2 ', 1.38000E-3, 1.11600E-2,   0.02470,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    48,     63,'1 ', 8.30000E-4, 1.88400E-2,   1.66668,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    49,     56,'1 ', 3.70000E-4, 3.66000E-3,   0.00830,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    49,     57,'1 ', 5.50000E-4, 5.86000E-3,   0.01246,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    51,     62,'1 ', 1.07000E-2, 7.90500E-2,   0.36670,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    52,     62,'1 ', 1.07000E-2, 7.90500E-2,   0.36670,    0.00,    0.00,  ␣
```

(continues on next page)

(continued from previous page)

```
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    53,     56,'1 ', 7.30000E-4, 1.02500E-2,   0.02558,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    53,     56,'2 ', 7.30000E-4, 1.02500E-2,   0.02558,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    54,     57,'1 ', 1.19000E-3, 1.24400E-2,   0.02798,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    54,     57,'2 ', 1.19000E-3, 1.24400E-2,   0.02798,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    54,     61,'1 ', 1.28000E-3, 9.79000E-3,   0.02120,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    56,     57,'1 ', 1.10000E-3, 1.18900E-2,   0.02514,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    60,    136,'1 ', 1.40000E-3, 2.64000E-2,   0.10200,    0.00,    0.00,␣
↪3070.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    60,    149,'1 ', 6.50000E-4, 1.18700E-2,   0.04672,    0.00,    0.00,␣
↪3070.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    60,    149,'2 ', 6.50000E-4, 1.18700E-2,   0.04672,    0.00,    0.00,␣
↪3070.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    63,    138,'1 ', 1.79000E-3, 2.52400E-2,   0.53546,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    63,    138,'2 ', 1.79000E-3, 2.52400E-2,   0.53546,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    63,    141,'1 ', 2.00000E-4, 4.10000E-3,   0.29620,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    65,     77,'1 ', 7.00000E-4, 7.40000E-2,   4.87000,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    67,     70,'1 ', 6.00000E-5, 1.31000E-3,   0.00378,    0.00,    0.00,␣
↪3020.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    67,     70,'2 ', 6.00000E-5, 1.16000E-3,   0.00332,    0.00,    0.00,␣
↪3020.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    68,     71,'1 ', 1.00000E-5, 3.00000E-4,   0.01434,    0.00,    0.00,␣
↪3450.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    68,     71,'2 ', 1.00000E-5, 3.00000E-4,   0.01844,    0.00,    0.00,␣
↪3450.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    68,     75,'1 ', 2.30000E-4, 4.51000E-3,   0.33320,    0.00,    0.00,␣
↪2175.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    68,     75,'2 ', 2.00000E-4, 4.46000E-3,   0.30500,    0.00,    0.00,␣
↪2175.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    72,     74,'1 ', 1.79000E-3, 1.40500E-2,   3.68000,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    73,     77,'1 ', 1.13000E-3, 2.06900E-2,   1.85526,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    74,     75,'1 ', 1.96000E-3, 3.30400E-2,   1.88000,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    74,     77,'1 ', 1.42000E-3, 2.25800E-2,   1.88000,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    75,     77,'1 ', 1.20000E-3, 2.31600E-2,   1.71520,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    75,     77,'2 ', 3.00000E-4, 2.00000E-2,   3.60000,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    75,     81,'1 ', 6.30000E-4, 1.41200E-2,   1.09756,    0.00,    0.00,␣
```

(continues on next page)

(continued from previous page)

```
↪3450.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
   75,     81,'2 ', 1.09000E-3, 2.40800E-2,   1.55542,    0.00,    0.00,␣
↪3020.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
   75,     81,'3 ', 1.08000E-3, 2.40900E-2,   1.55348,    0.00,    0.00,␣
↪3020.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
   77,     79,'1 ', 2.00000E-4, 8.20000E-3,   1.30000,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
   77,     79,'2 ', 2.00000E-4, 8.20000E-3,   1.30000,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
   80,     98,'1 ', 2.64000E-3, 5.35600E-2,   5.29066,    0.00,    0.00,␣
↪3600.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
   80,    179,'1 ',-0.00000E+0,-2.66700E-2,   0.00000,    0.00,    0.00,␣
↪1732.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
   81,     86,'1 ', 4.10000E-4, 7.37000E-3,   0.72694,    0.00,    0.00,␣
↪3450.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
   81,     90,'1 ', 6.60000E-4, 1.26600E-2,   0.95976,    0.00,    0.00,␣
↪3020.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
   81,     94,'1 ', 6.60000E-4, 1.26600E-2,   0.95976,    0.00,    0.00,␣
↪3020.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
   82,     88,'1 ', 7.20000E-4, 1.38200E-2,   1.27572,    0.00,    0.00,␣
↪3600.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
   82,     93,'1 ', 7.80000E-4, 1.50200E-2,   1.13810,    0.00,    0.00,␣
↪3020.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
   82,     97,'1 ', 7.40000E-4, 1.41300E-2,   1.06634,    0.00,    0.00,␣
↪3020.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
   82,    167,'1 ',-0.00000E+0,-7.20000E-3,   0.00000,    0.00,    0.00,␣
↪1800.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
   82,    169,'1 ',-0.00000E+0,-8.64000E-3,   0.00000,    0.00,    0.00,␣
↪1800.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
   82,    171,'1 ',-0.00000E+0,-1.00000E-2,   0.00000,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
   83,     98,'1 ',-0.00000E+0,-2.66700E-2,   0.00000,    0.00,    0.00,␣
↪1732.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
   84,    155,'1 ', 6.20000E-3, 6.73000E-2,   1.11560,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
   85,     87,'1 ', 6.00000E-4, 1.03600E-2,   1.01456,    0.00,    0.00,␣
↪3450.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
   85,     89,'1 ', 1.20000E-4, 2.38000E-3,   0.21926,    0.00,    0.00,␣
↪3600.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
   85,    179,'1 ', 1.22000E-3, 2.37300E-2,   2.20710,    0.00,    0.00,␣
↪3600.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
   86,     87,'1 ',-0.00000E+0,-1.26300E-2,   0.00000,    0.00,    0.00,␣
↪2000.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
   88,     89,'1 ',-0.00000E+0,-8.58000E-3,   0.00000,    0.00,    0.00,␣
↪2000.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
   90,     91,'1 ',-0.00000E+0,-1.26300E-2,   0.00000,    0.00,    0.00,␣
↪2400.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
   91,     92,'1 ', 7.40000E-4, 1.42800E-2,   1.08220,    0.00,    0.00,␣
↪3020.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
   92,     93,'1 ',-0.00000E+0,-1.26300E-2,   0.00000,    0.00,    0.00,␣
↪2400.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
   94,     95,'1 ',-0.00000E+0,-1.26300E-2,   0.00000,    0.00,    0.00,␣
```

(continues on next page)

(continued from previous page)

```
↪2000.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    95,     96,'1 ', 7.40000E-4, 1.42800E-2,   1.08220,    0.00,    0.00,␣
↪3020.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    96,     97,'1 ',-0.00000E+0,-1.26300E-2,   0.00000,    0.00,    0.00,␣
↪2000.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
    99,    100,'1 ', 2.48200E-2, 1.69380E-1,   0.20232,    0.00,    0.00, ␣
↪838.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
    99,    116,'1 ', 1.48000E-2, 1.01010E-1,   0.12066,    0.00,    0.00, ␣
↪838.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
   100,    104,'1 ', 1.38200E-2, 9.26800E-2,   0.11060,    0.00,    0.00, ␣
↪747.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
   100,    105,'1 ', 1.66800E-2, 1.13810E-1,   0.13608,    0.00,    0.00, ␣
↪838.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
   100,    112,'1 ', 1.11300E-2, 6.67800E-2,   0.07286,    0.00,    0.00, ␣
↪752.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
   100,    112,'2 ', 1.05000E-2, 6.54000E-2,   0.06860,    0.00,    0.00, ␣
↪602.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
   100,    112,'3 ', 1.10500E-2, 6.64200E-2,   0.07160,    0.00,    0.00, ␣
↪752.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
   100,    116,'1 ', 3.90300E-2, 2.74030E-1,   0.31072,    0.00,    0.00, ␣
↪747.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
   101,    103,'1 ', 7.90000E-4, 1.93700E-2,   1.32850,    0.00,    0.00,   ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
   101,    107,'1 ', 8.70000E-4, 2.08700E-2,   1.45710,    0.00,    0.00,   ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
   101,    107,'2 ', 8.70000E-4, 2.08700E-2,   1.45710,    0.00,    0.00,   ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
   103,    106,'1 ', 8.30000E-4, 1.98500E-2,   0.00000,    0.00,    0.00,␣
↪2450.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
   103,    133,'1 ', 9.30000E-4, 3.64400E-2,   1.38950,    0.00,    0.00,␣
↪2450.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
   103,    134,'1 ', 7.20000E-4, 1.60000E-2,   1.08790,    0.00,    0.00,␣
↪2450.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
   104,    116,'1 ', 3.05800E-2, 2.04600E-1,   0.24472,    0.00,    0.00, ␣
↪747.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
   105,    116,'1 ', 2.23500E-2, 1.61060E-1,   0.18342,    0.00,    0.00, ␣
↪838.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
   106,    107,'1 ', 1.53000E-3, 1.47000E-2,   0.00000,    0.00,    0.00,␣
↪1560.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
   106,    109,'1 ', 5.30000E-4, 1.29700E-2,   0.00000,    0.00,    0.00,␣
↪2450.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
   107,    132,'1 ', 2.00000E-5,-1.33100E-2,   0.00000,    0.00,    0.00,␣
↪1800.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
   107,    134,'1 ', 2.00000E-5,-9.98000E-3,   0.00000,    0.00,    0.00,␣
↪2450.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
   107,    173,'1 ',-0.00000E+0,-9.35000E-3,   0.00000,    0.00,    0.00,␣
↪2134.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
   107,    175,'1 ',-0.00000E+0,-9.44000E-3,   0.00000,    0.00,    0.00,␣
↪2134.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
   107,    177,'1 ',-0.00000E+0,-9.35000E-3,   0.00000,    0.00,    0.00,␣
↪2134.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
   110,    119,'1 ',-0.00000E+0,-1.00000E-2,   0.00000,    0.00,    0.00,␣
```

(continues on next page)

(continued from previous page)

```
↪2667.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
  110,    172,'1 ',-0.00000E+0,-1.00000E-2,   0.00000,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  113,    123,'1 ', 2.00000E-5,-9.98000E-3,   0.00000,    0.00,    0.00,␣
↪1800.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
  113,    125,'1 ', 2.00000E-5,-9.98000E-3,   0.00000,    0.00,    0.00,␣
↪1800.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
  113,    168,'1 ',-0.00000E+0,-7.20000E-3,   0.00000,    0.00,    0.00,␣
↪1800.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  113,    170,'1 ',-0.00000E+0,-7.20000E-3,   0.00000,    0.00,    0.00,␣
↪1800.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  114,    124,'1 ', 1.00000E-5,-6.66000E-3,   0.00000,    0.00,    0.00,␣
↪1800.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  114,    126,'1 ', 1.00000E-5,-6.66000E-3,   0.00000,    0.00,    0.00,␣
↪1800.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  114,    127,'1 ', 1.00000E-5,-1.12000E-2,   0.00000,    0.00,    0.00,␣
↪2667.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
  114,    129,'1 ', 1.00000E-5,-7.20000E-3,   0.00000,    0.00,    0.00,␣
↪2667.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
  118,    122,'1 ',-0.00000E+0,-1.00000E-2,   0.00000,    0.00,    0.00,␣
↪2667.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  118,    128,'1 ', 1.00000E-5,-1.12000E-2,   0.00000,    0.00,    0.00,␣
↪2667.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  118,    130,'1 ', 1.00000E-5,-3.60000E-3,   0.00000,    0.00,    0.00,␣
↪2667.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  118,    131,'1 ', 1.00000E-5,-7.55000E-3,   0.00000,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
  118,    133,'1 ', 1.00000E-5,-1.09800E-2,   0.00000,    0.00,    0.00,␣
↪2450.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
  119,    120,'1 ', 7.60000E-4, 1.95200E-2,   1.82450,    0.00,    0.00,␣
↪2894.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
  120,    121,'1 ',-0.00000E+0,-1.00000E-2,   0.00000,    0.00,    0.00,␣
↪2667.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
  121,    122,'1 ', 8.20000E-4, 2.11900E-2,   1.98420,    0.00,    0.00,␣
↪2894.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
  123,    124,'1 ', 1.40000E-3, 2.33800E-2,   1.47500,    0.00,    0.00,␣
↪2396.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
  125,    126,'1 ', 1.40000E-3, 2.33800E-2,   1.47500,    0.00,    0.00,␣
↪2396.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
  127,    128,'1 ', 1.54000E-3, 3.40900E-2,   2.31140,    0.00,    0.00,␣
↪2450.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
  129,    130,'1 ', 9.50000E-4, 2.10200E-2,   1.42520,    0.00,    0.00,␣
↪2450.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
  131,    132,'1 ', 1.65000E-3, 5.71900E-2,   2.47740,    0.00,    0.00,␣
↪2450.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
  135,    151,'1 ', 4.20000E-4, 9.05000E-3,   0.66794,    0.00,    0.00,␣
↪3600.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
  136,    142,'1 ', 1.90000E-3, 2.58000E-2,   0.09840,    0.00,    0.00,␣
↪2320.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
  136,    149,'1 ', 8.45000E-3, 7.03400E-2,   0.15954,    0.00,    0.00,␣
↪1160.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
  138,    141,'1 ', 1.93000E-3, 2.77900E-2,   4.67120,    0.00,    0.00,␣
```

(continues on next page)

(continued from previous page)

```
↪3600.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
  138,    146,'1 ', 5.60000E-4, 1.41500E-2,   1.04290,    0.00,    0.00,␣
↪3600.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  140,    142,'1 ', 1.10000E-3, 1.27000E-2,   0.04800,    0.00,    0.00,␣
↪2320.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
  141,    144,'1 ', 2.80000E-4, 7.53000E-3,   0.51736,    0.00,    0.00,␣
↪3600.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
  141,    144,'2 ', 3.50000E-4, 7.50000E-3,   0.55360,    0.00,    0.00,␣
↪3600.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
  141,    146,'1 ', 1.90000E-3, 3.10000E-2,   4.14020,    0.00,    0.00,␣
↪3600.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
  141,    150,'1 ', 6.00000E-4, 1.28000E-2,   0.94620,    0.00,    0.00,␣
↪3600.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
  141,    152,'1 ', 4.40000E-4, 1.12500E-2,   0.82920,    0.00,    0.00,␣
↪3600.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
  141,    152,'2 ', 4.40000E-4, 1.12500E-2,   0.82920,    0.00,    0.00,␣
↪3600.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
  142,    145,'1 ', 1.38000E-3, 5.39900E-2,   0.15252,    0.00,    0.00,␣
↪2320.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  142,    153,'1 ', 3.20000E-3, 3.95000E-2,   0.14400,    0.00,    0.00,␣
↪2320.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  144,    150,'1 ', 2.10000E-4, 4.57000E-3,   0.32336,    0.00,    0.00,␣
↪3600.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
  149,    153,'1 ', 2.85000E-3, 3.64900E-2,   0.12656,    0.00,    0.00,␣
↪2320.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
  149,    153,'2 ', 1.38000E-3, 3.39900E-2,   0.11252,    0.00,    0.00,␣
↪2320.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
  150,    151,'1 ', 4.00000E-4, 9.30000E-3,   0.68560,    0.00,    0.00,␣
↪3600.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
  152,    174,'1 ',-0.00000E+0,-9.35000E-3,   0.00000,    0.00,    0.00,␣
↪2134.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  152,    176,'1 ',-0.00000E+0,-9.35000E-3,   0.00000,    0.00,    0.00,␣
↪2134.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  152,    178,'1 ',-0.00000E+0,-8.40000E-3,   0.00000,    0.00,    0.00,␣
↪2100.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  154,    155,'1 ', 2.40000E-3, 3.32000E-2,   0.58490,    0.00,    0.00,   ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,2,   0.00,   1,1.0000
  154,    157,'1 ', 5.20000E-3, 6.02000E-2,   1.01000,    0.00,    0.00,   ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
  154,    157,'2 ', 4.90000E-3, 5.37000E-2,   0.88430,    0.00,    0.00,   ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
  154,    159,'1 ', 1.70000E-3, 2.25000E-2,   0.39920,    0.00,    0.00,   ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
  154,    159,'2 ', 2.10000E-3, 2.38000E-2,   0.38450,    0.00,    0.00,   ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
  154,    164,'1 ', 1.20000E-3, 1.72000E-2,   0.29870,    0.00,    0.00,   ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
  154,    166,'1 ', 8.00000E-4, 1.06000E-2,   0.20390,    0.00,    0.00,   ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
  155,    166,'1 ', 1.60000E-3, 2.26000E-2,   0.38100,    0.00,    0.00,   ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
  156,    160,'1 ', 1.08000E-2, 9.65000E-2,   0.32960,    0.00,    0.00,   ␣
```

(continues on next page)

(continued from previous page)

```
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
   157,    163,'1 ', 9.60000E-3, 8.78000E-2,   1.42650,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
   157,    164,'1 ', 3.40000E-3, 3.92000E-2,   0.65240,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
   157,    165,'1 ', 3.40000E-3, 3.74000E-2,   0.62080,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
   157,    165,'2 ', 3.40000E-3, 3.72000E-2,   0.61820,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
   159,    165,'1 ', 3.80000E-3, 3.40000E-2,   0.58240,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
   159,    165,'2 ', 3.20000E-3, 3.49000E-2,   0.57220,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
   167,    168,'1 ', 1.03000E-3, 2.33800E-2,   1.58040,    0.00,    0.00,␣
↪3020.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
   169,    170,'1 ', 1.07000E-3, 2.47000E-2,   1.52700,    0.00,    0.00,␣
↪3020.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
   171,    172,'1 ', 1.03000E-3, 3.23000E-2,   2.79600,    0.00,    0.00,  ␣
↪0.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
   173,    174,'1 ', 1.23000E-3, 2.65900E-2,   1.98702,    0.00,    0.00,␣
↪3600.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
   175,    176,'1 ', 1.23000E-3, 2.66200E-2,   1.98880,    0.00,    0.00,␣
↪3600.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
   177,    178,'1 ', 1.12000E-3, 2.51700E-2,   1.83586,    0.00,    0.00,␣
↪3600.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,   0.00,   1,1.0000
 0 /End of Branch data, Begin Transformer data
     1,     2,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'                  ␣
↪                     ',1,   1,1.0000
 0.00000E+0, 1.46000E-2,   100.00
1.00000,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.50000,␣
↪0.51000, 1.50000, 0.51000, 159, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
     1,     3,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'                  ␣
↪                     ',1,   1,1.0000
 0.00000E+0, 1.73000E-2,   100.00
0.95450,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.50000,␣
↪0.51000, 1.50000, 0.51000, 159, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
     4,     5,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'                  ␣
↪                     ',1,   1,1.0000
 0.00000E+0, 1.23800E-2,   100.00
1.00000,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.50000,␣
↪0.51000, 1.50000, 0.51000, 159, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
     4,    10,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'                  ␣
↪                     ',1,   1,1.0000
 0.00000E+0, 1.50000E-2,   100.00
1.00000,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.50000,␣
↪0.51000, 1.50000, 0.51000, 159, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
     6,     7,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'                  ␣
↪                     ',1,   1,1.0000
```

(continues on next page)

(continued from previous page)

```
 0.00000E+0, 1.10000E-2,   100.00
1.06300,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.50000,␣
↪0.51000, 1.50000, 0.51000, 159, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
     6,     7,     0,'2 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'                  ␣
↪                     ',1,   1,1.0000
 0.00000E+0, 1.10000E-2,   100.00
1.06300,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.50000,␣
↪0.51000, 1.50000, 0.51000, 159, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
     7,     8,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'                  ␣
↪                     ',1,   1,1.0000
 0.00000E+0, 5.90000E-3,   100.00
1.00000,   0.000,   0.000,     0.00,     0.00,  3000.00, 0,      0, 1.50000,␣
↪0.51000, 1.50000, 0.51000, 159, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
     7,     9,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'                  ␣
↪                     ',1,   1,1.0000
 2.80000E-4, 1.38000E-2,   100.00
1.00000,   0.000,   0.000,     0.00,     0.00,   430.00, 0,      0, 1.50000,␣
↪0.51000, 1.50000, 0.51000, 159, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
     7,     9,     0,'2 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'                  ␣
↪                     ',1,   1,1.0000
 2.90000E-4, 1.39000E-2,   100.00
1.00000,   0.000,   0.000,     0.00,     0.00,   430.00, 0,      0, 1.50000,␣
↪0.51000, 1.50000, 0.51000, 159, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
    11,    12,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'                  ␣
↪                     ',1,   1,1.0000
 0.00000E+0, 6.66000E-3,   100.00
1.08000,   0.000,   0.000,     0.00,     0.00,  2000.00, 0,      0, 1.50000,␣
↪0.51000, 1.50000, 0.51000, 159, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
    15,    14,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'                  ␣
↪                     ',1,   1,1.0000
 6.00000E-5, 4.95000E-3,   100.00
1.10610,   0.000,   0.000,     0.00,     0.00,  3066.00, 0,      0, 1.50000,␣
↪0.51000, 1.50000, 0.51000, 159, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
    16,    17,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'                  ␣
↪                     ',1,   1,1.0000
 0.00000E+0, 6.00000E-3,   100.00
1.04350,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.50000,␣
↪0.51000, 1.50000, 0.51000, 159, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
    30,    29,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'                  ␣
↪                     ',1,   1,1.0000
 0.00000E+0, 1.50000E-3,   100.00
1.05000,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.50000,␣
↪0.51000, 1.50000, 0.51000, 159, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
```

(continues on next page)

(continued from previous page)

```
    31,    32,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'                    ␣
↪                     ',1,   1,1.0000
 0.00000E+0, 1.00000E-2,   100.00
1.10000,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.50000,␣
↪0.51000, 1.50000, 0.51000, 159, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
    33,    34,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'                    ␣
↪                     ',1,   1,1.0000
 0.00000E+0, 2.00000E-3,   100.00
1.00000,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.50000,␣
↪0.51000, 1.50000, 0.51000, 159, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
    84,    35,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'                    ␣
↪                     ',1,   1,1.0000
 0.00000E+0, 4.60000E-3,   100.00
1.00000,   0.000,   0.000,     0.00,     0.00,  2000.00, 0,      0, 1.50000,␣
↪0.51000, 1.50000, 0.51000, 159, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
    37,    47,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'                    ␣
↪                     ',1,   1,1.0000
 1.30000E-4, 6.93000E-3,   100.00
1.05000,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.50000,␣
↪0.51000, 1.50000, 0.51000, 159, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
    38,    39,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'                    ␣
↪                     ',1,   1,1.0000
 5.00000E-4, 2.38000E-2,   100.00
1.00000,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.50000,␣
↪0.51000, 1.50000, 0.51000, 159, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
    41,    42,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'                    ␣
↪                     ',1,   1,1.0000
 5.80000E-4, 2.53500E-2,   100.00
1.04910,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.50000,␣
↪0.51000, 1.50000, 0.51000, 159, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
    43,    44,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'                    ␣
↪                     ',1,   1,1.0000
 0.00000E+0, 5.20000E-3,   100.00
1.02500,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.50000,␣
↪0.51000, 1.50000, 0.51000, 159, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
    47,    46,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'                    ␣
↪                     ',1,   1,1.0000
 4.99000E-3, 1.14730E-1,   100.00
1.04780,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.50000,␣
↪0.51000, 1.50000, 0.51000, 159, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
    48,    47,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'                    ␣
↪                     ',1,   1,1.0000
 2.60000E-4, 1.38600E-2,   100.00
1.05000,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.50000,␣
```

(continues on next page)

(continued from previous page)

```
↪0.51000, 1.50000, 0.51000, 159, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
    51,    50,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'                  ␣
↪                      ',1,   1,1.0000
 5.90000E-4, 1.49100E-2,   100.00
1.00170,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.50000,␣
↪0.51000, 1.50000, 0.51000, 159, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
    52,    50,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'                  ␣
↪                      ',1,   1,1.0000
 5.90000E-4, 1.49100E-2,   100.00
1.00170,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.50000,␣
↪0.51000, 1.50000, 0.51000, 159, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
    53,    50,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'                  ␣
↪                      ',1,   1,1.0000
 3.00000E-4, 1.33000E-2,   100.00
1.00000,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.50000,␣
↪0.51000, 1.50000, 0.51000, 159, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
    53,    50,     0,'2 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'                  ␣
↪                      ',1,   1,1.0000
 3.00000E-4, 1.34000E-2,   100.00
1.00000,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.50000,␣
↪0.51000, 1.50000, 0.51000, 159, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
    55,    54,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'                  ␣
↪                      ',1,   1,1.0000
 1.30000E-4, 1.38600E-2,   100.00
1.01060,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.50000,␣
↪0.51000, 1.50000, 0.51000, 159, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
    55,    54,     0,'2 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'                  ␣
↪                      ',1,   1,1.0000
 1.30000E-4, 1.38600E-2,   100.00
1.01060,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.50000,␣
↪0.51000, 1.50000, 0.51000, 159, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
    59,    60,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'                  ␣
↪                      ',1,   1,1.0000
 0.00000E+0, 1.15000E-3,   100.00
1.01330,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.50000,␣
↪0.51000, 1.50000, 0.51000, 159, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
    63,    62,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'                  ␣
↪                      ',1,   1,1.0000
 2.00000E-4, 2.33800E-2,   100.00
0.97890,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.50000,␣
↪0.51000, 1.50000, 0.51000, 159, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
    65,    64,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'                  ␣
↪                      ',1,   1,1.0000
```

(continues on next page)

(continued from previous page)

```
 0.00000E+0, 5.00000E-3,   100.00
1.09000,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.50000,␣
↪0.51000, 1.50000, 0.51000, 159, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
    67,    66,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'                  ␣
↪                     ',1,   1,1.0000
 8.90000E-4, 2.99000E-2,   100.00
0.98730,   0.000,   0.000,     0.00,     0.00,   250.00, 0,      0, 1.50000,␣
↪0.51000, 1.50000, 0.51000, 159, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
    68,    67,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'                  ␣
↪                     ',1,   1,1.0000
 2.00000E-4, 1.18100E-2,   100.00
1.02380,   0.000,   0.000,     0.00,     0.00,  1008.00, 0,      0, 1.50000,␣
↪0.51000, 1.50000, 0.51000, 159, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
    68,    67,     0,'2 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'                  ␣
↪                     ',1,   1,1.0000
 9.00000E-5, 7.35000E-3,   100.00
1.02380,   0.000,   0.000,     0.00,     0.00,  1300.00, 0,      0, 1.50000,␣
↪0.51000, 1.50000, 0.51000, 159, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
    67,    69,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'                  ␣
↪                     ',1,   1,1.0000
 0.00000E+0, 1.03400E-2,   100.00
1.04550,   0.000,   0.000,     0.00,     0.00,  2000.00, 0,      0, 1.50000,␣
↪0.51000, 1.50000, 0.51000, 159, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
    71,    70,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'                  ␣
↪                     ',1,   1,1.0000
 0.00000E+0, 2.21000E-3,   100.00
1.02340,   0.000,   0.000,     0.00,     0.00,  2500.00, 0,      0, 1.50000,␣
↪0.51000, 1.50000, 0.51000, 159, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
    75,    76,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'                  ␣
↪                     ',1,   1,1.0000
 0.00000E+0, 3.75000E-3,   100.00
1.09770,   0.000,   0.000,     0.00,     0.00,  5000.00, 0,      0, 1.50000,␣
↪0.51000, 1.50000, 0.51000, 159, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
    79,    78,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'                  ␣
↪                     ',1,   1,1.0000
 0.00000E+0, 2.50000E-3,   100.00
1.06600,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.50000,␣
↪0.51000, 1.50000, 0.51000, 159, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
    83,    84,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'                  ␣
↪                     ',1,   1,1.0000
 0.00000E+0, 7.20000E-3,   100.00
1.05000,   0.000,   0.000,     0.00,     0.00,  1500.00, 0,      0, 1.50000,␣
↪0.51000, 1.50000, 0.51000, 159, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
```

(continues on next page)

(continued from previous page)

```
   101,   102,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'                  ␣
↪                     ',1,   1,1.0000
 0.00000E+0, 9.80000E-3,   100.00
1.05000,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.50000,␣
↪0.51000, 1.50000, 0.51000, 159, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
   108,   107,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'                  ␣
↪                     ',1,   1,1.0000
 3.00000E-4, 1.74000E-2,   100.00
1.11900,   0.000,   0.000,     0.00,     0.00,   840.00, 0,      0, 1.50000,␣
↪0.51000, 1.50000, 0.51000, 159, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
   108,   107,     0,'2 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'                  ␣
↪                     ',1,   1,1.0000
 2.00000E-4, 1.19000E-2,   100.00
1.11900,   0.000,   0.000,     0.00,     0.00,  1120.00, 0,      0, 1.50000,␣
↪0.51000, 1.50000, 0.51000, 159, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
   111,   112,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'                  ␣
↪                     ',1,   1,1.0000
 0.00000E+0, 2.28100E-2,   100.00
0.91740,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.50000,␣
↪0.51000, 1.50000, 0.51000, 159, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
   112,   113,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'                  ␣
↪                     ',1,   1,1.0000
 1.00000E-4, 1.74000E-2,   100.00
1.11900,   0.000,   0.000,     0.00,     0.00,   840.00, 0,      0, 1.50000,␣
↪0.51000, 1.50000, 0.51000, 159, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
   115,   116,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'                  ␣
↪                     ',1,   1,1.0000
 0.00000E+0, 1.81500E-2,   100.00
0.90910,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.50000,␣
↪0.51000, 1.50000, 0.51000, 159, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
   116,   118,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'                  ␣
↪                     ',1,   1,1.0000
 2.00000E-4, 1.25000E-2,   100.00
1.11900,   0.000,   0.000,     0.00,     0.00,  1120.00, 0,      0, 1.50000,␣
↪0.51000, 1.50000, 0.51000, 159, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
   117,   118,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'                  ␣
↪                     ',1,   1,1.0000
 0.00000E+0, 4.48000E-3,   100.00
0.94520,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.50000,␣
↪0.51000, 1.50000, 0.51000, 159, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
   137,   138,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'                  ␣
↪                     ',1,   1,1.0000
 0.00000E+0, 1.51200E-2,   100.00
0.99600,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.50000,␣
```

(continues on next page)

(continued from previous page)

```
↪0.51000, 1.50000, 0.51000, 159, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
   139,   140,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'                  ␣
↪                      ',1,   1,1.0000
 0.00000E+0, 3.65000E-3,   100.00
0.97870,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.50000,␣
↪0.51000, 1.50000, 0.51000, 159, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
   143,   144,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'                  ␣
↪                      ',1,   1,1.0000
 0.00000E+0, 5.16000E-3,   100.00
0.98430,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.50000,␣
↪0.51000, 1.50000, 0.51000, 159, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
   144,   145,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'                  ␣
↪                      ',1,   1,1.0000
 0.00000E+0, 5.00000E-3,   100.00
1.00000,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.50000,␣
↪0.51000, 1.50000, 0.51000, 159, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
   146,   147,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'                  ␣
↪                      ',1,   1,1.0000
 0.00000E+0, 9.80000E-3,   100.00
1.05000,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.50000,␣
↪0.51000, 1.50000, 0.51000, 159, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
   148,   149,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'                  ␣
↪                      ',1,   1,1.0000
 0.00000E+0, 1.02600E-2,   100.00
0.98710,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.50000,␣
↪0.51000, 1.50000, 0.51000, 159, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
   152,   153,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'                  ␣
↪                      ',1,   1,1.0000
 0.00000E+0, 1.14900E-2,   100.00
1.06310,   0.000,   0.000,     0.00,     0.00,  1120.00, 0,      0, 1.50000,␣
↪0.51000, 1.50000, 0.51000, 159, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
   152,   153,     0,'2 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'                  ␣
↪                      ',1,   1,1.0000
 0.00000E+0, 1.14900E-2,   100.00
1.06310,   0.000,   0.000,     0.00,     0.00,  1120.00, 0,      0, 1.50000,␣
↪0.51000, 1.50000, 0.51000, 159, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
   152,   153,     0,'3 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'                  ␣
↪                      ',1,   1,1.0000
 0.00000E+0, 1.14900E-2,   100.00
1.06310,   0.000,   0.000,     0.00,     0.00,  1120.00, 0,      0, 1.50000,␣
↪0.51000, 1.50000, 0.51000, 159, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
   155,   156,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'                  ␣
↪                      ',1,   1,1.0000
```

(continues on next page)

(continued from previous page)

```
 3.00000E-4, 1.81000E-2,   100.00
1.00000,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.50000,␣
↪0.51000, 1.50000, 0.51000, 159, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
   157,   158,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'                  ␣
↪                     ',1,   1,1.0000
 2.00000E-4, 5.80000E-3,   100.00
0.98550,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.50000,␣
↪0.51000, 1.50000, 0.51000, 159, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
   160,   161,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'                  ␣
↪                     ',1,   1,1.0000
 5.00000E-4, 1.41000E-2,   100.00
1.05880,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.50000,␣
↪0.51000, 1.50000, 0.51000, 159, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
   163,   162,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'                  ␣
↪                     ',1,   1,1.0000
 0.00000E+0, 1.95000E-2,   100.00
1.00000,   0.000,   0.000,     0.00,     0.00,     0.00, 0,      0, 1.50000,␣
↪0.51000, 1.50000, 0.51000, 159, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
 0 /End of Transformer data, Begin Area interchange data
 0 /End of Area interchange data, Begin Two-terminal dc line data
 0 /End of Two-terminal dc line data, Begin VSC dc line data
 0 /End of VSC dc line data, Begin Impedance correction table data
 0 /End of Impedance correction table data, Begin Multi-terminal dc line data
 0 /End of Multi-terminal dc line data, Begin Multi-section line data
 0 /End of Multi-section line data, Begin Zone data
   1,'1           '
 0 /End of Zone data, Begin Inter-area transfer data
 0 /End of Inter-area transfer data, Begin Owner data
   1,'1           '
 0 /End of Owner data, Begin FACTS device data
 0 /End of FACTS device data, Begin Switched shunt data
 0 /End of Switched shunt data, Begin GNE device data
 0 /End of GNE device data
Q
```

Dynamic Data (PSS/E DYR format)

```
data = !cat wecc.dyr
print('\n'.join(data))
```

```
      3 'GENROU' 1     3.9000      0.32000E-01  0.54000      0.62000E-01
          2.6400      5.0000       1.8600       1.7800      0.25000
         0.45300     0.19500      0.14500       1.9714       6.9000      /
      3 'IEEEST' 1          3            0       0.0000       0.0000
          0.0000      0.0000       0.0000       0.0000       0.0000
```

(continues on next page)

(continued from previous page)

```
        0.0000       0.0000      0.75000       1.0000       4.2000
       -2.0000      0.10000     -0.10000       0.0000       0.0000      /
     3 'ESST3A' 1    0.20000E-01  0.30000     -0.20000       8.0000
        1.0000       5.0000       20.000       0.0000       99.000
       -99.000       1.0000       3.6700      0.43500       6.4800
       0.10000E-01  0.98000E-02   4.8600       3.3300      0.40000
        99.000       0.0000    /
     3 'IEEEG1' 1          0           0        20.000      0.10000
        0.0000      0.20000       1.0000      -1.0000      0.95000
        0.0000      0.10000       0.0000       0.0000       0.0000
        0.0000       0.0000       0.0000      0.30000       0.0000
        8.7200      0.70000       0.0000    /
     5 'GENROU' 1     4.0000      0.32000E-01  0.52000      0.63000E-01
        2.6100       0.0000       1.8100       1.7300      0.24500
       0.51000      0.19000      0.14000       1.7850       7.1400      /
     5 'ST2CUT' 1          1            0            0            0
        5.5000       0.0000      0.30000E-01   0.0000       10.000
        10.000      0.40000      0.40000E-01   0.0000       0.0000
        0.0000       0.0000      0.50000E-01 -0.50000E-01   0.0000
        0.0000      /
     5 'ESST3A' 1    0.20000E-01  0.20000     -0.20000       8.2000
        1.0000       5.0000       50.000       0.0000       99.000
       -99.000       1.0000       3.5600      0.33500       5.8100
       0.10000E-01  0.77000E-02   5.6000       3.7500      0.40000
        99.000       0.0000    /
     5 'IEEEG1' 1          0           0        20.000      0.10000
        0.0000      0.20000       1.0000      -1.0000      0.95000
        0.0000      0.10000       0.0000       0.0000       0.0000
        0.0000       0.0000       0.0000      0.30000       0.0000
        8.7200      0.70000       0.0000    /
     8 'GENROU' 1     8.4000      0.35000E-01  0.46000      0.70000E-01
        3.4200       0.0000       1.7600       1.5800      0.28500
       0.48500      0.20500      0.15500       1.8429       6.4500      /
     8 'ST2CUT' 1          1            0            0            0
        10.000       0.0000       0.0000       0.0000       3.0000
        3.0000      0.15000      0.50000E-01  0.15000      0.50000E-01
       0.15000      0.50000E-01  0.50000E-01 -0.50000E-01   0.0000
        0.0000      /
     8 'EXST1'  1    0.20000E-01   99.000      -99.000       0.0000
       0.20000E-01   50.000      0.20000E-01   9999.0      -9999.0
        0.0000      0.10000E-01   1.0000    /
     8 'IEEEG1' 1          0           0        20.000      0.10000
        0.0000      0.20000       1.0000      -1.0000      0.95000
        0.0000      0.10000       0.0000       0.0000       0.0000
        0.0000       0.0000       0.0000      0.30000       0.0000
        8.7200      0.70000       0.0000    /
    10 'GENROU' 1     4.5000      0.40000E-01  0.50000      0.60000E-01
        2.4500       0.0000       1.7000       1.6500      0.25000
       0.50000      0.19000      0.14000       1.7850       7.1400      /
    10 'ST2CUT' 1          1            0            0            0
        0.0000       0.0000      0.25000E-01   0.0000      0.16000E-01
       0.16000E-01  0.26000      0.26000E-01  0.26000      0.26000E-01
```

(continues on next page)

(continued from previous page)

```
     0.0000       0.0000      0.10000     -0.10000       0.0000
     0.0000      /
10 'ESDC2A' 1    0.20000E-01   50.000      0.50000E-01  0.20000E-01
     0.0000       3.0000      -3.0000       0.0000      0.51200
    0.70000E-01   1.3000       0.0000       3.9825      0.50000
     5.3100       1.0490    /
10 'IEEEG1' 1          0           0        20.000      0.10000
     0.0000      0.20000       1.0000      -1.0000      0.95000
     0.0000      0.10000       0.0000       0.0000       0.0000
     0.0000       0.0000       0.0000      0.30000       0.0000
     8.7200      0.70000       0.0000    /
12 'GENROU' 1     4.2000      0.32000E-01  0.60000      0.66000E-01
     3.5900       0.0000       1.7600       1.6800      0.22000
    0.40000      0.17500      0.13000       1.9225       7.6900      /
12 'ST2CUT' 1          1            0            0            0
     10.000       0.0000       0.0000       0.0000       3.0000
     3.0000      0.15000      0.50000E-01  0.15000      0.50000E-01
    0.15000      0.50000E-01  0.50000E-01 -0.50000E-01   0.0000
     0.0000      /
12 'EXST1'  1    0.20000E-01   99.000      -99.000       1.0000
     10.000       75.000      0.20000E-01   9999.0      -9999.0
     0.0000      0.10000E-01   1.0000    /
12 'IEEEG1' 1          0           0        20.000      0.10000
     0.0000      0.20000       1.0000      -1.0000      0.95000
     0.0000      0.10000       0.0000       0.0000       0.0000
     0.0000       0.0000       0.0000      0.30000       0.0000
     8.7200      0.70000       0.0000    /
14 'GENROU' 1     8.2000      0.33000E-01  0.48000      0.55000E-01
     3.8300       0.0000       1.9800       1.8500      0.37500
    0.56700      0.28500      0.22500       1.4800       4.4400      /
14 'ST2CUT' 1          1            0            0            0
     10.000       0.0000       0.0000       0.0000       3.0000
     3.0000      0.15000      0.50000E-01  0.15000      0.50000E-01
    0.15000      0.50000E-01  0.50000E-01 -0.50000E-01   0.0000
     0.0000      /
14 'ESST3A' 1    0.20000E-01  0.20000     -0.20000       6.8000
     1.0000       5.0000       50.000       0.0000       99.000
    -99.000       1.0000       3.7700      0.12400       6.5200
    0.10000E-01  0.50000E-02   6.7700       0.0000      0.40000
     99.000       0.0000    /
14 'IEEEG1' 1          0           0        20.000      0.10000
     0.0000      0.20000       1.0000      -1.0000      0.95000
     0.0000      0.10000       0.0000       0.0000       0.0000
     0.0000       0.0000       0.0000      0.30000       0.0000
     8.7200      0.70000       0.0000    /
17 'GENROU' 1     3.8000      0.31000E-01  0.49000      0.58000E-01
     2.6300       0.0000       1.6700      0.61000      0.26000
    0.45300      0.20500      0.14500       1.9714       6.9000      /
17 'ST2CUT' 1          1            0            0            0
     0.0000       0.0000       0.0000       0.0000       30.000
     30.000      0.23000      0.25000E-01  0.23000      0.25000E-01
     0.0000       0.0000      0.60000E-01 -0.60000E-01   0.0000
```

(continues on next page)

(continued from previous page)

```
     0.0000      /
17 'ESDC2A' 1    0.20000E-01   25.000      0.20000E-01  0.20000E-01
     0.0000       3.0000      -3.0000       0.0000      0.64200
    0.10300      0.99400       0.0000       2.5950      0.35600
     3.4600       1.0000    /
17 'IEEEG1' 1          0           0        20.000      0.10000
     0.0000      0.20000       1.0000      -1.0000      0.95000
     0.0000      0.10000       0.0000       0.0000       0.0000
     0.0000       0.0000       0.0000      0.30000       0.0000
     8.7200      0.70000       0.0000    /
29 'GENROU' 1     8.0000      0.50000E-01  0.20000      0.80000E-01
     4.3400       0.0000      0.90000      0.60000      0.25000
    0.60000      0.15000      0.90000E-01   2.2000       11.000      /
29 'ST2CUT' 1          3            0            0            0
    -3.5000       0.0000       0.0000       0.0000       1.0000
     1.0000       0.0000       3.0000      0.80000E-01  0.27000E-01
    0.80000E-01  0.27000E-01  0.10000     -0.10000       0.0000
     0.0000      /
29 'EXST1'  1    0.20000E-01   99.000      -99.000       0.0000
    0.20000E-01   72.500      0.20000E-01   9999.0      -9999.0
     0.0000      0.10000E-01   1.5000    /
29 'IEEEG1' 1          0           0        20.000      0.10000
     0.0000      0.20000       1.0000      -1.0000      0.95000
     0.0000      0.10000       0.0000       0.0000       0.0000
     0.0000       0.0000       0.0000      0.30000       0.0000
     8.7200      0.70000       0.0000    /
34 'GENROU' 1     4.6000      0.40000E-01  0.37000      0.10000
     2.9500       0.0000       1.8250       1.7800      0.29100
    0.44000      0.25000      0.23900       1.3933       4.1800      /
34 'ST2CUT' 1          1            0            0            0
     10.000       0.0000      0.10000       0.0000       1.0000
     1.0000      0.20000      0.10000E-01   0.0000       0.0000
     0.0000       0.0000      0.50000E-01 -0.50000E-01   0.0000
     0.0000      /
34 'EXST1'  1    0.20000E-01   99.000      -99.000       0.0000
    0.20000E-01   50.000      0.80000E-01   9999.0      -9999.0
     0.0000      0.15000E-01   1.0000    /
34 'IEEEG1' 1          0           0        20.000      0.10000
     0.0000      0.20000       1.0000      -1.0000      0.95000
     0.0000      0.10000       0.0000       0.0000       0.0000
     0.0000       0.0000       0.0000      0.30000       0.0000
     8.7200      0.70000       0.0000    /
35 'GENROU' 1     4.1000      0.33000E-01  0.56000      0.62000E-01
     2.3200       0.0000       2.0700       1.9900      0.28000
    0.49000      0.21500      0.15500       1.8429       6.4500      /
35 'ST2CUT' 1          1            0            0            0
     10.000       0.0000       0.0000       0.0000       3.0000
     3.0000      0.15000      0.50000E-01  0.15000      0.50000E-01
    0.15000      0.50000E-01  0.50000E-01 -0.50000E-01   0.0000
     0.0000      /
35 'EXST1'  1    0.20000E-01   99.000      -99.000       0.0000
    0.20000E-01   50.000      0.36000       9999.0      -9999.0
```

(continues on next page)

(continued from previous page)

```
     0.0000      0.30000E-01   1.0000    /
35 'IEEEG1' 1          0           0        20.000      0.10000
     0.0000      0.20000       1.0000      -1.0000      0.95000
     0.0000      0.10000       0.0000       0.0000       0.0000
     0.0000       0.0000       0.0000      0.30000       0.0000
     8.7200      0.70000       0.0000    /
39 'GENROU' 1     9.8000      0.28000E-01  0.20000      0.16000
     6.4100       0.0000      0.99500      0.56800      0.19500
    0.56800      0.10000      0.89000E-01   2.2400       11.200      /
39 'ST2CUT' 1          1            0            0            0
     0.0000       0.0000       0.0000       0.0000       15.000
     15.000       0.0000      0.53000E-01   0.0000      0.53000E-01
     0.0000       0.0000      0.50000E-01 -0.50000E-01   0.0000
     0.0000      /
39 'EXST1'  1    0.20000E-01   99.000      -99.000       0.0000
    0.20000E-01   75.000      0.16000       9999.0      -9999.0
     0.0000      0.10000E-01   1.0000    /
39 'IEEEG1' 1          0           0        20.000      0.10000
     0.0000      0.30000       1.0000      -1.0000      0.95000
     0.0000      0.10000       0.0000       0.0000       0.0000
     0.0000       0.0000       0.0000      0.30000       0.0000
     10.000      0.70000       0.0000    /
42 'GENROU' 1     4.8000      0.37000E-01  0.50000      0.75000E-01
     4.1300       0.0000       1.7000       1.6200      0.25600
    0.41000      0.10000      0.85000E-01   1.8429       6.4500      /
42 'ST2CUT' 1          3            0            0            0
     0.0000       0.0000      0.30000E-01   0.0000       1.5000
     1.5000      0.37500       12.000       0.0000       0.0000
     0.0000       0.0000      0.50000E-01 -0.50000E-01   0.0000
     0.0000      /
42 'ESDC2A' 1    0.20000E-01   20.000      0.20000      0.20000E-01
     0.0000       2.0000      -2.0000       0.0000      0.54500
    0.90000E-01   1.0000       0.0000       3.0975      0.41000
     4.1300       1.0000    /
42 'IEEEG1' 1          0           0        20.000      0.80000E-01
     0.0000      0.15000       1.0000      -1.0000      0.95000
     0.0000      0.50000E-01   0.0000       0.0000       0.0000
     0.0000       0.0000       0.0000      0.28000       0.0000
     10.000      0.72000       0.0000    /
44 'GENROU' 1     4.7000      0.31000E-01  0.67000      0.61000E-01
     2.8800       0.0000       1.9900       1.9000      0.24500
    0.42000      0.10000      0.85000E-01   1.9057       6.6700      /
44 'ST2CUT' 1          1            0            0            0
     3.0000       0.0000       0.0000       0.0000       10.000
     10.000      0.30000      0.30000E-01  0.30000      0.30000E-01
     0.0000       0.0000      0.10000     -0.10000       0.0000
     0.0000      /
44 'ESST3A' 1    0.20000E-01  0.20000     -0.20000       7.2000
     1.0000       4.0000       50.000       0.0000       99.000
    -99.000       1.0000       4.3500      0.24300       8.4000
    0.10000E-01  0.50000E-02   6.3600       20.000      0.60000
     99.000       0.0000    /
```

(continues on next page)

(continued from previous page)

```
    44 'IEEEG1' 1          0           0        20.000      0.10000
         0.0000      0.20000       1.0000      -1.0000      0.95000
         0.0000      0.10000       0.0000       0.0000       0.0000
         0.0000       0.0000       0.0000      0.30000       0.0000
         8.7200      0.70000       0.0000    /
    46 'GENROU' 1     7.4000      0.30000E-01  0.20000      0.16000
         3.8000       0.0000      0.61900      0.36500      0.23800
        0.36500      0.10000      0.85000E-01   1.7850       7.1400      /
    46 'ST2CUT' 1          1            0           0            0
         10.000       0.0000       0.0000       0.0000       3.0000
         3.0000      0.15000      0.50000E-01  0.15000      0.50000E-01
        0.15000      0.50000E-01  0.50000E-01 -0.50000E-01   0.0000
         0.0000      /
    46 'EXST1'  1    0.20000E-01   99.000      -99.000       0.0000
        0.20000E-01   50.000      0.16000       9999.0      -9999.0
         0.0000      0.15000E-01   1.0000    /
    46 'IEEEG1' 1          0           0        20.000      0.90000E-01
         0.0000      0.20000       1.0000      -1.0000      0.98000
         0.0000      0.30000      0.10000       0.0000       0.0000
         0.0000       0.0000       0.0000       0.0000       0.0000
         0.0000      0.90000       0.0000    /
    64 'GENROU' 1     4.7800      0.41000E-01  0.53000      0.67000E-01
         3.3800       0.0000       1.6026       1.5818      0.28525
        0.45122      0.23079      0.16596       1.9475       7.7900      /
    64 'ST2CUT' 1          1            0           0            0
         3.0000       0.0000      0.30000E-01   0.0000       20.000
         20.000       6.7200      0.67200       0.0000       0.0000
         0.0000       0.0000      0.10000     -0.10000       0.0000
         0.0000      /
    64 'EXST1'  1    0.20000E-01   99.000      -99.000       0.0000
        0.20000E-01   47.500      0.13000       9999.0      -9999.0
         0.0000      0.20000E-01   1.0000    /
    64 'IEEEG1' 1          0           0        20.000      0.10000
         0.0000      0.20000       1.0000      -1.0000      0.95000
         0.0000      0.50000       0.0000       0.0000       0.0000
         0.0000       0.0000       0.0000      0.30000       0.0000
         8.7200      0.70000       0.0000    /
    69 'GENROU' 1     4.0000      0.20000E-01  0.20000      0.50000E-01
         3.0000       0.0000       1.4846      0.83403      0.61720
        0.83403      0.47870      0.40033       1.3900       4.1700      /
    69 'ST2CUT' 1          1            0           0            0
         10.000       0.0000       0.0000       0.0000       3.0000
         3.0000      0.15000      0.50000E-01  0.15000      0.50000E-01
        0.15000      0.50000E-01  0.50000E-01 -0.50000E-01   0.0000
         0.0000      /
    69 'EXST1'  1    0.20000E-01   99.000      -99.000       0.0000
        0.20000E-01   50.000      0.16000       9999.0      -9999.0
         0.0000      0.26000E-01   1.0000    /
    69 'IEEEG1' 1          0           0        20.000      0.10000
         0.0000      0.20000       1.0000      -1.0000      0.95000
         0.0000      0.10000      0.30000       0.0000       0.0000
         0.0000       0.0000       0.0000      0.30000       0.0000
```

(continues on next page)

(continued from previous page)

```
     8.7200      0.40000       0.0000    /
 76 'GENROU' 1     8.0000      0.30000E-01  0.20000      0.60000E-01
     3.6700       0.0000       1.2009      0.84062      0.46099
    0.84062      0.32024      0.30991       1.3900       4.1700      /
 76 'ST2CUT' 1          3            0            0            0
   -0.40000E-01   0.0000       0.0000       0.0000       29.500
     29.500       1.5000      0.15000       1.5000      0.15000
     0.0000       0.0000      0.90000E-01 -0.90000E-01   0.0000
     0.0000      /
 76 'ESDC2A' 1    0.20000E-01   20.000      0.60000E-01  0.20000E-01
     0.0000       3.0000      -3.0000       0.0000      0.13000
    0.50000E-01   1.3000       0.0000       3.3750      0.30000
     4.5000       1.0000    /
 76 'IEEEG1' 1          0           0        20.000      0.10000
     0.0000      0.20000       1.0000      -1.0000       1.0500
     0.0000      0.80000       0.0000       0.0000       0.0000
     0.0000       0.0000       0.0000      0.30000       0.0000
     8.7200      0.70000       0.0000    /
 78 'GENROU' 1     7.2000      0.43000E-01  0.20000      0.79000E-01
     3.4600       0.0000      0.63800      0.42400      0.22800
    0.42400      0.16100      0.12700       1.9675       7.8700      /
 78 'ST2CUT' 1          3            0            0            0
   -0.60000E-01   0.0000       0.0000       0.0000       29.500
     29.500       1.5000      0.15000       1.5000      0.15000
     0.0000       0.0000      0.90000E-01 -0.90000E-01   0.0000
     0.0000      /
 78 'ESDC2A' 1    0.20000E-01   20.000      0.10000      0.20000E-01
     0.0000       3.0000      -3.0000       0.0000      0.57000
    0.50000E-01   1.2000       0.0000       3.3750      0.30000
     4.5000       1.0000    /
 78 'IEEEG1' 1          0           0        20.000      0.10000
     0.0000      0.20000       1.0000      -1.0000      0.95000
     0.0000      0.10000       0.0000       0.0000       0.0000
     0.0000       0.0000       0.0000      0.30000       0.0000
     8.7200      0.70000       0.0000    /
102 'GENROU' 1     6.1200      0.52000E-01   1.5000      0.14400
     3.4600       0.0000       2.1290       2.0740      0.46700
     1.2700      0.31100      0.25000       1.3333       4.0000      /
102 'ST2CUT' 1          1            0            0            0
     10.000       0.0000       0.0000       0.0000       3.0000
     3.0000      0.15000      0.50000E-01  0.15000      0.50000E-01
    0.15000      0.50000E-01  0.50000E-01 -0.50000E-01   0.0000
     0.0000      /
102 'EXST1'  1    0.20000E-01   99.000      -99.000      0.20000E-01
    0.20000E-01   40.000      0.20000E-01   9999.0      -9999.0
     0.0000      0.10000E-01   1.0000    /
102 'IEEEG1' 1          0           0        20.000      0.10000
     0.0000      0.20000       1.0000      -1.0000      0.95000
     0.0000      0.10000       0.0000       0.0000       0.0000
     0.0000       0.0000       0.0000      0.30000       0.0000
     8.7200      0.70000       0.0000    /
111 'GENROU' 1     4.0000      0.51000E-01  0.20000      0.33000E-01
```

(continues on next page)

(continued from previous page)

```
      4.3900       5.0000       2.0548       1.3111      0.62622
      1.3111      0.51468      0.27397       1.7850       7.1400      /
111 'IEEEST' 1          3            0       0.0000       0.0000
      0.0000       0.0000       0.0000       0.0000       0.0000
      0.0000       0.0000      0.75000       1.0000       4.2000
     -2.0000      0.10000     -0.10000       0.0000       0.0000      /
111 'ESDC2A' 1    0.20000E-01   40.500      0.60000E-01  0.20000E-01
      0.0000       2.0000      -2.0000       0.0000      0.50000
     0.50000E-01   1.0000       0.0000       2.6250      0.54000
      20.000       1.0000    /
111 'IEEEG1' 1          0            0       20.000      0.10000
      0.0000      0.20000       1.0000      -1.0000      0.95000
      0.0000      0.10000      0.10000       0.0000       0.0000
      0.0000       0.0000       0.0000      0.30000       0.0000
      8.7200      0.60000       0.0000    /
115 'GENROU' 1     7.8000      0.60000E-01  0.20000      0.11000
      4.1600       0.0000      0.75000      0.53000      0.32000
     0.53000      0.30000      0.21500       1.5500       4.6500      /
115 'IEEEST' 1          3            0       0.0000       0.0000
      0.0000       0.0000       0.0000       0.0000       0.0000
      0.0000       0.0000      0.75000       1.0000       4.2000
     -2.0000      0.10000     -0.10000       0.0000       0.0000      /
115 'EXST1'  1    0.20000E-01   99.000      -99.000       0.0000
     0.20000E-01   40.000      0.20000E-01   9999.0      -9999.0
      0.0000      0.10000E-01   1.2000    /
115 'IEEEG1' 1          0            0       20.000      0.10000
      0.0000      0.20000       1.0000      -1.0000      0.95000
      0.0000      0.10000       0.0000       0.0000       0.0000
      0.0000       0.0000       0.0000      0.30000       0.0000
      8.7200      0.70000       0.0000    /
117 'GENROU' 1     4.5000      0.40000E-01  0.50000      0.60000E-01
      3.8200       0.0000       1.7000       1.6500      0.25000
     0.50000      0.19000      0.14000       1.7850       7.1400      /
117 'IEEEST' 1          3            0       0.0000       0.0000
      0.0000       0.0000       0.0000       0.0000       0.0000
      0.0000       0.0000      0.75000       1.0000       4.2000
     -2.0000      0.10000     -0.10000       0.0000       0.0000      /
117 'ESDC2A' 1    0.20000E-01   40.000      0.50000E-01  0.20000E-01
      0.0000       3.0000      -3.0000       0.0000      0.51200
     0.70000E-01   1.3000       0.0000       3.9825      0.50000
      5.3100       1.0490    /
117 'IEEEG1' 1          0            0       20.000      0.10000
      0.0000      0.20000       1.0000      -1.0000      0.95000
      0.0000      0.10000       0.0000       0.0000       0.0000
      0.0000       0.0000       0.0000      0.30000       0.0000
      8.7200      0.70000       0.0000    /
137 'GENROU' 1     7.8000      0.60000E-01  0.20000      0.11000
      6.0700       0.0000      0.75000      0.53000      0.32000
     0.53000      0.30000      0.21500       1.5500       4.6500      /
137 'ST2CUT' 1          1            0            0            0
      10.000       0.0000       0.0000       0.0000       3.0000
      3.0000      0.15000      0.50000E-01  0.15000      0.50000E-01
```

(continues on next page)

(continued from previous page)

```
     0.15000      0.50000E-01  0.50000E-01 -0.50000E-01   0.0000
      0.0000      /
137 'EXST1'  1    0.20000E-01   99.000      -99.000       0.0000
     0.20000E-01   50.000      0.20000E-01   9999.0      -9999.0
      0.0000      0.10000E-01   1.2000    /
137 'IEEEG1' 1          0           0        20.000      0.10000
      0.0000      0.20000       1.0000      -1.0000      0.95000
      0.0000      0.10000       0.0000       0.0000       0.0000
      0.0000       0.0000       0.0000      0.30000       0.0000
      8.7200      0.70000       0.0000    /
139 'GENROU' 1     5.3000      0.35000E-01  0.50000      0.77000E-01
      2.8200       0.0000       1.6200       1.5700      0.24500
     0.37600      0.10000      0.95000E-01   2.1060       10.530      /
139 'ST2CUT' 1          3            0            0            0
      0.0000       0.0000       0.0000       0.0000       29.500
      29.500       1.5000      0.15000       1.5000      0.15000
      0.0000       0.0000      0.90000E-01 -0.90000E-01   0.0000
      0.0000      /
139 'ESDC2A' 1    0.20000E-01   20.000      0.50000E-01  0.20000E-01
      0.0000       3.0000      -3.0000       0.0000      0.60000
     0.70000E-01   1.3000       0.0000       2.8200      0.32000
      3.7600       1.0000    /
139 'IEEEG1' 1          0           0        20.000      0.10000
      0.0000      0.20000       1.0000      -1.0000      0.95000
      0.0000      0.10000       0.0000       0.0000       0.0000
      0.0000       0.0000       0.0000      0.30000       0.0000
      8.7200      0.70000       0.0000    /
143 'GENROU' 1     8.0000      0.50000E-01   1.5000      0.60000E-01
      2.7500       0.0000       1.9500       1.8700      0.34000
      1.0000      0.29000      0.24000       1.3900       4.1700      /
143 'ST2CUT' 1          1            0            0            0
     0.50000       0.0000      0.42000E-01   0.0000       10.000
      10.000       1.0000      0.80000E-01   1.7000      0.80000E-01
      0.0000       0.0000      0.50000E-01 -0.50000E-01   0.0000
      0.0000      /
143 'EXST1'  1    0.20000E-01   99.000      -99.000       0.0000
     0.20000E-01   50.000      0.19000       9999.0      -9999.0
      0.0000      0.15000E-01   1.0000    /
143 'IEEEG1' 1          0           0        20.000      0.10000
      0.0000      0.20000       1.0000      -1.0000      0.95000
      0.0000      0.10000       0.0000       0.0000       0.0000
      0.0000       0.0000       0.0000      0.30000       0.0000
      8.7200      0.70000       0.0000    /
147 'GENROU' 1     8.4000      0.35000E-01  0.46000      0.70000E-01
      3.4900       0.0000       1.7600       1.5800      0.28500
     0.48500      0.20500      0.15500       1.8429       6.4500      /
147 'ST2CUT' 1          1            0            0            0
      5.0000       0.0000      0.29000E-01   0.0000       2.4000
      2.4000      0.14400      0.40000E-01  0.37700      0.10500
      0.0000       0.0000      0.50000E-01 -0.50000E-01   0.0000
      0.0000      /
147 'EXST1'  1    0.20000E-01   99.000      -99.000       0.0000
```

(continues on next page)

(continued from previous page)

```
     0.20000E-01   40.000      0.20000E-01   9999.0      -9999.0
      0.0000      0.10000E-01   1.0000    /
147 'IEEEG1' 1          0           0        20.000      0.10000
      0.0000      0.20000       1.0000      -1.0000      0.95000
      0.0000      0.10000       0.0000       0.0000       0.0000
      0.0000       0.0000       0.0000      0.30000       0.0000
      8.7200      0.70000       0.0000    /
148 'GENROU' 1     4.3000      0.32000E-01  0.43000      0.66000E-01
      3.0300       0.0000       1.7900       1.7150      0.22000
     0.44000      0.10000      0.85000E-01   1.8525       7.4100      /
148 'ST2CUT' 1          1            0            0            0
      10.000       0.0000       0.0000       0.0000       3.0000
      3.0000      0.15000      0.50000E-01  0.15000      0.50000E-01
     0.15000      0.50000E-01  0.50000E-01 -0.50000E-01   0.0000
      0.0000      /
148 'EXST1'  1    0.20000E-01   99.000      -99.000       0.0000
     0.20000E-01   40.000      0.20000       9999.0      -9999.0
      0.0000      0.35000E-01   1.7500    /
148 'IEEEG1' 1          0           0        20.000      0.10000
      0.0000      0.20000       1.0000      -1.0000      0.95000
      0.0000      0.10000       0.0000       0.0000       0.0000
      0.0000       0.0000       0.0000      0.30000       0.0000
      8.7200      0.70000       0.0000    /
158 'GENROU' 1     3.9000      0.32000E-01  0.49000      0.62000E-01
      2.8200       0.0000       1.6400       1.5700      0.24000
     0.43000      0.18500      0.13000       1.9225       7.6900      /
158 'ST2CUT' 1          1            0            0            0
      8.0800       0.0000      0.30000E-01   0.0000       10.000
      10.000      0.15000      0.50000E-01  0.15000      0.50000E-01
      0.0000       0.0000      0.50000E-01 -0.50000E-01   0.0000
      0.0000      /
158 'EXST1'  1    0.20000E-01   99.000      -99.000       1.0000
      10.000       40.000      0.20000E-01   9999.0      -9999.0
      0.0000      0.10000E-01   1.0000    /
158 'IEEEG1' 1          0           0        20.000      0.10000
      0.0000      0.20000       1.0000      -1.0000      0.95000
      0.0000      0.10000       0.0000       0.0000       0.0000
      0.0000       0.0000       0.0000      0.30000       0.0000
      8.7200      0.70000       0.0000    /
161 'GENROU' 1     5.2100      0.42000E-01   1.5000      0.42000E-01
      3.0100       0.0000       1.7980       1.7780      0.32400
     0.45000      0.26000      0.19300       1.4800       5.1800      /
161 'ST2CUT' 1          1            0            0            0
      10.000       0.0000      0.30000E-01   0.0000       30.000
      30.000      0.40000      0.50000E-01  0.40000      0.50000E-01
      0.0000       0.0000      0.40000E-01 -0.40000E-01   0.0000
      0.0000      /
161 'EXST1'  1    0.20000E-01   99.000      -99.000       0.0000
     0.20000E-01   50.000      0.10000       9999.0      -9999.0
      0.0000      0.12000E-01   1.0000    /
161 'IEEEG1' 1          0           0        20.000      0.10000
      0.0000      0.20000       1.0000      -1.0000      0.95000
```

(continues on next page)

(continued from previous page)

```
        0.0000      0.10000       0.0000       0.0000       0.0000
        0.0000       0.0000       0.0000      0.30000       0.0000
        8.7200      0.70000       0.0000    /
```

PYTHON MODULE INDEX

a

andes.cli, 938
andes.interop, 929
andes.interop.gridcal, 930
andes.interop.matpower, 931
andes.interop.pandapower, 933
andes.interop.pypowsybl, 937
andes.io, 920
andes.io.json, 922
andes.io.matpower, 923
andes.io.psse, 925
andes.io.streaming, 926
andes.io.txt, 927
andes.io.xlsx, 927
andes.main, 939
andes.plot, 916
andes.routines, 897
andes.routines.adaptive, 898
andes.routines.base, 900
andes.routines.cpf, 901
andes.routines.criteria, 903
andes.routines.daeint, 904
andes.routines.eig, 907
andes.routines.pflow, 909
andes.routines.qndf, 910
andes.routines.se, 913
andes.routines.tds, 914
andes.system, 880
andes.system.codegen, 880
andes.system.config_runtime, 882
andes.system.dae_compactor, 883
andes.system.facade, 884
andes.system.helpers, 888
andes.system.registry, 890
andes.utils.paths, 948
andes.utils.snapshot, 951
andes.utils.widgets, 952
andes.variables, 891
andes.variables.dae, 891
andes.variables.fileman, 895
andes.variables.report, 896