

---

# **ANDES Manual**

***Release 1.8.10.post29+g97528c6***

**Hantao Cui**

**Dec 14, 2023**



# ANDES MANUAL

<b>1 Getting started</b>	<b>3</b>
1.1 Package Overview . . . . .	3
1.2 Installation . . . . .	4
1.2.1 New to Python . . . . .	4
1.2.2 Extra support package . . . . .	5
1.2.3 Develop Install . . . . .	5
1.2.4 Updating ANDES . . . . .	7
1.2.5 Uninstall Multiple Copies . . . . .	7
1.2.6 Troubleshooting . . . . .	7
1.3 Tutorial . . . . .	8
1.3.1 Command line . . . . .	8
1.3.2 Scripting . . . . .	20
1.3.3 Cheatsheet . . . . .	28
1.3.4 Documentation . . . . .	28
1.4 Config . . . . .	29
1.4.1 Format . . . . .	29
1.4.2 Adjusting on the fly . . . . .	30
1.4.3 Config reference . . . . .	31
1.5 Input formats . . . . .	34
1.5.1 ANDES xlsx . . . . .	34
1.5.2 PSS/E RAW and DYSR . . . . .	38
1.5.3 MATPOWER . . . . .	42
1.5.4 ANDES JSON . . . . .	42
1.5.5 Disturbances . . . . .	44
1.6 Test Cases . . . . .	45
1.6.1 Summary . . . . .	45
1.6.2 Example data . . . . .	46
1.6.3 MATPOWER cases . . . . .	46
1.6.4 How to contribute . . . . .	47
1.7 Performance . . . . .	47
1.7.1 Numba compilation . . . . .	47
1.7.2 MATPOWER cases . . . . .	48
1.8 Verification . . . . .	50
1.8.1 IEEE 14-Bus Verification . . . . .	50
1.8.2 CURENT NPCC Verification . . . . .	60

1.8.3	CURRENT WECC Verification . . . . .	89
1.9	Miscellaneous . . . . .	131
1.9.1	Per Unit System . . . . .	131
1.9.2	Notes . . . . .	132
1.9.3	Profiling Import . . . . .	133
1.9.4	What won't work . . . . .	133
1.10	Frequently Asked Questions . . . . .	133
1.10.1	Program Startup . . . . .	133
1.10.2	General . . . . .	133
1.10.3	Modeling . . . . .	133
1.11	License . . . . .	134
1.11.1	GNU Public License v3 . . . . .	134
1.12	Quick install . . . . .	134
<b>2</b>	<b>Examples</b> . . . . .	<b>135</b>
2.1	Simulate and Plot . . . . .	135
2.1.1	Import and Setting the Verbosity Level . . . . .	135
2.1.2	Run Time-Domain Simulation . . . . .	136
2.1.3	Export and Plot Results . . . . .	137
2.1.4	Cleanup . . . . .	148
2.2	Working with Data . . . . .	149
2.2.1	Load System from an ANDES XLSX File . . . . .	149
2.2.2	Load System from PSS/E RAW and DYR Files . . . . .	151
2.2.3	Variables . . . . .	156
2.2.4	Altering Fault duration . . . . .	159
2.2.5	Cleanup . . . . .	161
2.3	Inspecting Models . . . . .	161
2.3.1	Inspect Model Equations . . . . .	161
2.3.2	Check model documentation . . . . .	162
2.4	Eigenvalue Analysis . . . . .	169
2.4.1	Run Eigenvalue Analysis . . . . .	169
2.4.2	Plotting in S-Domain . . . . .	170
2.4.3	Report . . . . .	170
2.4.4	Parameter Sweep and Root Loci Plot . . . . .	189
2.4.5	Cleanup . . . . .	190
2.5	Using CLI from Notebook . . . . .	190
2.5.1	The ! magic in iPython . . . . .	190
2.5.2	Set up on Windows . . . . .	190
2.5.3	Running Shell Commands . . . . .	190
2.5.4	Run a simulation . . . . .	191
2.5.5	Check the output lst file . . . . .	194
2.5.6	Plot and save to file . . . . .	198
2.5.7	Display image . . . . .	199
2.5.8	Using xargs for index lookup . . . . .	199
2.5.9	Cleanup . . . . .	200
2.6	Batch Processing - Generate Cases . . . . .	200
2.6.1	Create Cases in Batch . . . . .	200
2.6.2	Parallel Simulation . . . . .	201

2.7	Batch Processing - in Memory . . . . .	205
2.7.1	Batch Power Flow Calculation . . . . .	206
2.7.2	Batch Time-Domain Simulation . . . . .	207
2.8	Changing Setpoints . . . . .	217
2.8.1	Step 1: Case Setup . . . . .	218
2.8.2	Step 2: Set the First Stop Time . . . . .	219
2.8.3	Step 3: Run Simulation . . . . .	219
2.8.4	Step 4. Apply the auxiliary power setpoints to TGOV1.paux0.v . . . . .	219
2.8.5	Step 5: Set Another New Setpoints and New Ending TIme. . . . .	224
2.9	Load Frequency Control . . . . .	226
2.9.1	Tripping a Generator in the IEEE 14-Bus System . . . . .	227
2.9.2	Adjusting Load to Compensate for the Generation Loss . . . . .	230
2.10	Profile in Notebook . . . . .	234
2.10.1	Profiling with Python CProfiler . . . . .	234
2.10.2	Profiling with line_profiler. . . . .	237
2.10.3	Cleanup . . . . .	246
2.11	MATPOWER Interface . . . . .	246
2.11.1	Create an Octave/MATLAB instance . . . . .	246
2.11.2	Convert to MATPOWER . . . . .	246
2.11.3	Convert from MATPOWER . . . . .	248
2.11.4	Add dynamic data . . . . .	249
2.12	pandapower Interface . . . . .	250
2.12.1	Load case . . . . .	251
2.12.2	Convert to pandapower net . . . . .	252
2.12.3	Comapre Power Flow Results . . . . .	253
2.12.4	Generator dispatch based on OPF from pandapower . . . . .	256
2.13	pypowsybl Interface . . . . .	259
2.13.1	Imports . . . . .	259
2.13.2	Conversion . . . . .	259
2.13.3	Diagrams . . . . .	259
<b>3</b>	<b>Development</b> . . . . .	<b>261</b>
3.1	System . . . . .	261
3.1.1	Overview . . . . .	261
3.1.2	DAE Storage . . . . .	264
3.1.3	Model and DAE Values . . . . .	265
3.1.4	Calling Model Methods . . . . .	266
3.1.5	Configuration . . . . .	267
3.2	Group . . . . .	268
3.2.1	andes.models.group.GroupBase . . . . .	268
3.3	Models . . . . .	273
3.3.1	andes.core.model.ModelData . . . . .	274
3.3.2	andes.core.model.Model . . . . .	277
3.3.3	andes.core.model.ModelCache . . . . .	290
3.3.4	andes.core.model.ModelCall . . . . .	291
3.3.5	Cache . . . . .	292
3.3.6	Define Voltage Ratings . . . . .	293
3.3.7	Commonly Used Attributes in Models . . . . .	295

3.3.8	Attributes in <i>Model.cache</i>	296
3.3.9	Abstract Jacobian Storage	298
3.3.10	Concrete Jacobian Storage	298
3.4	Atomic Types	299
3.4.1	Value Provider	300
3.4.2	Equation Provider	300
3.5	Parameters	301
3.5.1	Background	301
3.6	Variables	324
3.6.1	andes.core.var.BaseVar	325
3.6.2	andes.core.var.ExtVar	327
3.6.3	andes.core.var.State	330
3.6.4	andes.core.var.Algeb	332
3.6.5	andes.core.var.ExtState	335
3.6.6	andes.core.var.ExtAlgeb	338
3.6.7	andes.core.var.AliasState	340
3.6.8	andes.core.var.AliasAlgeb	343
3.6.9	Variable, Equation and Address	346
3.6.10	Value and Equation Strings	346
3.6.11	Values Between DAE and Models	346
3.6.12	Flags for Value Overwriting	347
3.6.13	A <i>v_setter</i> Example	347
3.7	Services	347
3.7.1	andes.core.service BaseService	348
3.7.2	andes.core.service OperationService	349
3.7.3	Internal Constants	352
3.7.4	External Constants	354
3.7.5	Shape Manipulators	355
3.7.6	Value Manipulation	359
3.7.7	Idx and References	359
3.7.8	Events	362
3.7.9	Flags	363
3.7.10	Data Select	364
3.7.11	Miscellaneous	365
3.8	Discrete	367
3.8.1	Background	367
3.8.2	Limiters	371
3.8.3	Comparers	373
3.8.4	Deadband	375
3.8.5	Others	377
3.9	Blocks	379
3.9.1	Background	379
3.9.2	Transfer Functions	382
3.9.3	Saturation	400
3.9.4	Naming Convention	401
3.10	Examples	401
3.10.1	TGOV1	401
3.10.2	IEEEEST	404

<b>4</b>	<b>Release notes</b>	<b>409</b>
4.1	v1.9 Notes . . . . .	409
4.1.1	v1.9.0 (2023-12-07) . . . . .	409
4.2	v1.8 Notes . . . . .	409
4.2.1	v1.8.10 (2023-08-10) . . . . .	409
4.2.2	v1.8.9 (2023-06-22) . . . . .	410
4.2.3	v1.8.8 (2023-04-24) . . . . .	410
4.2.4	v1.8.7 (2023-03-10) . . . . .	410
4.2.5	v1.8.6 (2023-02-13) . . . . .	410
4.2.6	v1.8.5 (2022-12-22) . . . . .	410
4.2.7	v1.8.4 (2022-11-23) . . . . .	410
4.2.8	v1.8.3 (2022-11-15) . . . . .	411
4.2.9	v1.8.2 (2022-10-30) . . . . .	411
4.2.10	v1.8.1 (2022-09-24) . . . . .	411
4.2.11	v1.8.0 (2022-08-30) . . . . .	411
4.3	v1.7 Notes . . . . .	411
4.3.1	v1.7.8 (2022-08-24) . . . . .	411
4.3.2	v1.7.7 (2022-08-02) . . . . .	412
4.3.3	v1.7.6 (2022-07-11) . . . . .	412
4.3.4	v1.7.5 (2022-07-05) . . . . .	412
4.3.5	v1.7.4 (2022-07-01) . . . . .	412
4.3.6	v1.7.3 (2022-06-25) . . . . .	413
4.3.7	v1.7.2 (2022-06-07) . . . . .	413
4.3.8	v1.7.1 (2022-05-31) . . . . .	413
4.3.9	v1.7.0 (2022-05-22) . . . . .	413
4.4	v1.6 Notes . . . . .	414
4.4.1	v1.6.6 (2022-04-30) . . . . .	414
4.4.2	v1.6.5 (2022-04-19) . . . . .	414
4.4.3	v1.6.4 (2022-04-17) . . . . .	415
4.4.4	v1.6.3 (2022-04-06) . . . . .	415
4.4.5	v1.6.2 (2022-03-27) . . . . .	416
4.4.6	v1.6.1 (2022-03-13) . . . . .	416
4.4.7	v1.6.0 (2022-03-11) . . . . .	416
4.5	v1.5 Notes . . . . .	416
4.5.1	v1.5.12 (2022-03-05) . . . . .	416
4.5.2	v1.5.11 (2022-02-23) . . . . .	417
4.5.3	v1.5.10 (2022-02-01) . . . . .	417
4.5.4	v1.5.9 (2022-01-31) . . . . .	417
4.5.5	v1.5.8 (2021-12-21) . . . . .	417
4.5.6	v1.5.7 (2021-12-11) . . . . .	417
4.5.7	v1.5.6 (2021-11-25) . . . . .	418
4.5.8	v1.5.5 (2021-11-13) . . . . .	418
4.5.9	v1.5.4 (2021-11-02) . . . . .	418
4.5.10	v1.5.3 (2021-10-31) . . . . .	419
4.5.11	v1.5.2 (2021-10-27) . . . . .	419
4.5.12	v1.5.1 (2021-10-23) . . . . .	419
4.5.13	v1.5.0 (2021-10-13) . . . . .	419
4.6	v1.4 Notes . . . . .	420

4.6.1	v1.4.4 (2021-10-05) . . . . .	420
4.6.2	v1.4.3 (2021-09-25) . . . . .	420
4.6.3	v1.4.2 (2021-09-12) . . . . .	420
4.6.4	v1.4.1 (2021-09-12) . . . . .	420
4.6.5	v1.4.0 (2021-09-08) . . . . .	421
4.7	v1.3 Notes . . . . .	421
4.7.1	v1.3.12 (2021-08-22) . . . . .	421
4.7.2	v1.3.11 (2021-07-27) . . . . .	421
4.7.3	v1.3.10 (2021-06-08) . . . . .	421
4.7.4	v1.3.9 (2021-06-02) . . . . .	422
4.7.5	v1.3.8 (2021-06-02) . . . . .	422
4.7.6	v1.3.7 (2021-05-03) . . . . .	422
4.7.7	v1.3.6 (2021-04-23) . . . . .	422
4.7.8	v1.3.5 (2021-03-20) . . . . .	422
4.7.9	v1.3.4 (2021-03-13) . . . . .	422
4.7.10	v1.3.2 (2021-03-08) . . . . .	422
4.7.11	v1.3.1 (2021-03-07) . . . . .	423
4.7.12	v1.3.0 (2021-02-20) . . . . .	423
4.8	v1.2 Notes . . . . .	423
4.8.1	v1.2.9 (2021-01-16) . . . . .	423
4.8.2	v1.2.7 (2020-12-08) . . . . .	423
4.8.3	v1.2.6 (2020-12-01) . . . . .	424
4.8.4	v1.2.5 (2020-11-19) . . . . .	424
4.8.5	v1.2.4 (2020-11-13) . . . . .	424
4.8.6	v1.2.3 (2020-11-02) . . . . .	424
4.8.7	v1.2.2 (2020-11-01) . . . . .	424
4.8.8	v1.2.1 (2020-10-11) . . . . .	425
4.8.9	v1.2.0 (2020-10-10) . . . . .	425
4.9	v1.1 Notes . . . . .	425
4.9.1	v1.1.5 (2020-10-08) . . . . .	425
4.9.2	v1.1.4 (2020-09-22) . . . . .	426
4.9.3	v1.1.3 (2020-09-05) . . . . .	426
4.9.4	v1.1.2 (2020-09-03) . . . . .	426
4.9.5	v1.1.1 (2020-09-02) . . . . .	426
4.9.6	v1.1.0 (2020-09-01) . . . . .	426
4.10	v1.0 Notes . . . . .	427
4.10.1	v1.0.8 (2020-07-29) . . . . .	427
4.10.2	v1.0.7 (2020-07-18) . . . . .	427
4.10.3	v1.0.6 (2020-07-08) . . . . .	428
4.10.4	v1.0.5 (2020-07-02) . . . . .	428
4.10.5	v1.0.4 (2020-06-26) . . . . .	428
4.10.6	v1.0.3 (2020-06-02) . . . . .	428
4.10.7	v1.0.2 (2020-06-01) . . . . .	428
4.10.8	v1.0.1 (2020-05-27) . . . . .	428
4.10.9	v1.0.0 (2020-05-25) . . . . .	429
4.11	Pre-v1.0.0 . . . . .	429
4.11.1	v0.9.4 (2020-05-20) . . . . .	429
4.11.2	v0.9.3 (2020-05-05) . . . . .	429

4.11.3	v0.9.1 (2020-05-02) . . . . .	430
4.11.4	v0.8.8 (2020-04-28) . . . . .	430
4.11.5	v0.8.7 (2020-04-28) . . . . .	431
4.11.6	v0.8.6 (2020-04-21) . . . . .	431
4.11.7	v0.8.5 (2020-04-17) . . . . .	431
4.11.8	v0.8.4 (2020-04-07) . . . . .	432
4.11.9	v0.8.3 (2020-03-25) . . . . .	432
4.11.10	v0.8.0 (2020-02-12) . . . . .	432
4.11.11	v0.6.9 (2020-02-12) . . . . .	432
<b>5</b>	<b>Model reference</b>	<b>433</b>
5.1	ACLine . . . . .	434
5.1.1	Line . . . . .	434
5.2	ACShort . . . . .	437
5.2.1	Jumper . . . . .	437
5.3	ACTopology . . . . .	439
5.3.1	Bus . . . . .	439
5.4	Calculation . . . . .	440
5.4.1	ACE . . . . .	440
5.4.2	ACEc . . . . .	442
5.4.3	COI . . . . .	444
5.5	Collection . . . . .	445
5.5.1	Area . . . . .	446
5.6	DCLink . . . . .	446
5.6.1	Ground . . . . .	446
5.6.2	R . . . . .	447
5.6.3	L . . . . .	449
5.6.4	C . . . . .	450
5.6.5	RCp . . . . .	452
5.6.6	RCs . . . . .	453
5.6.7	RLs . . . . .	455
5.6.8	RLCs . . . . .	456
5.6.9	RLCp . . . . .	458
5.7	DCTopology . . . . .	459
5.7.1	Node . . . . .	459
5.8	DG . . . . .	461
5.8.1	PVD1 . . . . .	461
5.8.2	ESD1 . . . . .	469
5.8.3	EV1 . . . . .	476
5.8.4	EV2 . . . . .	483
5.9	DGProtection . . . . .	490
5.9.1	DGPRCT1 . . . . .	490
5.9.2	DGPRCTExt . . . . .	496
5.10	DataSeries . . . . .	502
5.10.1	TimeSeries . . . . .	502
5.11	DynLoad . . . . .	504
5.11.1	ZIP . . . . .	504
5.11.2	FLoad . . . . .	506

5.12	Exciter	507
5.12.1	EXDC2	507
5.12.2	IEEEEX1	512
5.12.3	ESDC1A	517
5.12.4	ESDC2A	523
5.12.5	EXST1	528
5.12.6	ESST3A	531
5.12.7	SEXS	538
5.12.8	IEEET1	540
5.12.9	EXAC1	545
5.12.10	EXAC2	551
5.12.11	EXAC4	558
5.12.12	ESST4B	561
5.12.13	AC8B	568
5.12.14	IEEET3	574
5.12.15	ESAC1A	579
5.12.16	ESST1A	585
5.12.17	ESAC5A	592
5.13	Experimental	598
5.14	FreqMeasurement	598
5.14.1	BusFreq	598
5.14.2	BusROCOF	600
5.15	Information	602
5.15.1	Summary	603
5.16	Interface	603
5.16.1	Fortescue	603
5.17	Motor	607
5.17.1	Motor3	607
5.17.2	Motor5	610
5.18	OutputSelect	614
5.18.1	Output	614
5.19	PLL	614
5.19.1	PLL1	615
5.19.2	PLL2	617
5.20	PSS	618
5.20.1	IEEEEST	619
5.20.2	ST2CUT	625
5.21	PhasorMeasurement	630
5.21.1	PMU	630
5.22	RenAerodynamics	631
5.22.1	WTARA1	631
5.22.2	WTARV1	633
5.23	RenExciter	634
5.23.1	REECA1	634
5.23.2	REECA1E	642
5.23.3	REECA1G	650
5.24	RenGen	658
5.24.1	REGCA1	659

5.24.2	REGCP1	664
5.24.3	REGCV1	671
5.24.4	REGCV2	676
5.24.5	REGF1	681
5.24.6	REGF2	686
5.24.7	REGF3	692
5.25	RenGovernor	699
5.25.1	WTDTA1	699
5.25.2	WTDS	702
5.26	RenPitch	705
5.26.1	WTPTA1	705
5.27	RenPlant	708
5.27.1	REPCA1	708
5.28	RenTorque	714
5.28.1	WTTQA1	714
5.29	StaticACDC	719
5.29.1	VSCShunt	719
5.30	StaticGen	722
5.30.1	PV	723
5.30.2	Slack	725
5.31	StaticLoad	728
5.31.1	PQ	728
5.32	StaticShunt	731
5.32.1	Shunt	731
5.32.2	ShuntTD	732
5.32.3	ShuntSw	733
5.33	SynGen	736
5.33.1	GENCLS	736
5.33.2	GENROU	741
5.33.3	PLBFU1	746
5.34	TimedEvent	749
5.34.1	Toggle	749
5.34.2	Fault	750
5.34.3	Alter	752
5.35	TurbineGov	754
5.35.1	TG2	754
5.35.2	TGOV1	757
5.35.3	TGOV1DB	760
5.35.4	TGOV1N	763
5.35.5	TGOV1NDB	766
5.35.6	IEEEG1	770
5.35.7	IEESGO	774
5.35.8	GAST	778
5.35.9	HYGOV	781
5.35.10	HYGOVDB	785
5.35.11	HYGOV4	789
5.36	Undefined	793
5.37	VoltComp	793

5.37.1	IEEEVC . . . . .	794
<b>6</b>	<b>API reference</b>	<b>797</b>
6.1	System . . . . .	797
6.1.1	andes.system . . . . .	797
6.1.2	andes.variables . . . . .	814
6.2	Routines . . . . .	829
6.2.1	andes.routines . . . . .	829
6.3	Plot . . . . .	850
6.3.1	andes.plot . . . . .	850
6.4	I/O . . . . .	861
6.4.1	andes.io . . . . .	862
6.5	Interoperability . . . . .	873
6.5.1	andes.interop . . . . .	873
6.6	Others . . . . .	882
6.6.1	andes.cli . . . . .	882
6.6.2	andes.main . . . . .	883
6.6.3	andes.utils.paths . . . . .	890
6.6.4	andes.utils.snapshot . . . . .	894
6.6.5	andes.utils.widgets . . . . .	895
	<b>Bibliography</b>	<b>897</b>
	<b>Python Module Index</b>	<b>899</b>
	<b>Index</b>	<b>901</b>

**Download documentation:** [PDF for stable version](#) | [PDF for development version](#)

**Useful Links:** [Binary Installer](#) | [Source Repository](#) | [Report Issues](#) | [Q&A](#) | [Try in Jupyter Notebooks](#) | [LTB Repository](#)



LTB ANDES is an open-source Python library for power system modeling, computation, analysis, and control, serving as the core simulation engine for the CURENT Large scale Testbed (LTB). It supports power flow calculation, transient stability simulation, and small-signal stability analysis for transmission systems. ANDES implements a symbolic-numeric framework for rapid prototyping of differential-algebraic equation-based models. In this framework, a comprehensive *library of models* is developed, including the full second-generation renewable models. Models in ANDES have been *verified* with commercial software.

## Getting started

New to ANDES? Check out the Getting Started guides. They contain tutorials to the ANDES command-line interface, scripting usages, as well as guides to configure ANDES and work with case files.

*[To the getting started guides](#)*

## Examples

The examples provide in-depth usage of ANDES in a Python scripting environment. Advanced usage and power system studies are shown with explanation.

*[To the examples](#)*

## Model development guide

Looking to implement new models, algorithms, and functionalities in ANDES? The development guide provides in-depth information on the design philosophy, data structure, and implementation of the hybrid symbolic-numeric framework.

*[To the development guide](#)*

## API reference

The API reference contains a detailed description of the ANDES package. The reference describes how the methods work and which parameters can be used. It assumes that you have an understanding of the key concepts.

*[To the API reference](#)*

## Using ANDES for Research?

Please cite our paper [Cui2021] if ANDES is used in your research for publication.



## GETTING STARTED

### 1.1 Package Overview

ANDES is an open-source Python package for power system modeling, computation, analysis, and control. It establishes a unique **hybrid symbolic-numeric framework** for modeling differential algebraic equations (DAEs) for numerical analysis. The main features of ANDES include

- a unique hybrid symbolic-numeric approach to modeling and simulation that enables descriptive DAE modeling and automatic numerical code generation
- a rich library of transfer functions and discontinuous components (including limiters, dead-bands, and saturation) available for prototyping models, which can be readily instantiated as multiple devices for system analysis
- industry-grade second-generation renewable models (solar PV, type 3 and type 4 wind), distributed PV, and energy storage model
- comes with the Newton method for power flow calculation, the implicit trapezoidal method for time-domain simulation, and full eigenvalue calculation
- rigorously verified models with commercial software. ANDES obtains identical time-domain simulation results for IEEE 14-bus and NPCC system with GENROU and multiple controller models. See the verification link for details.
- developed with performance in mind. While written in Python, ANDES comes with a performance package and can finish a 20-second transient simulation of a 2000-bus system in a few seconds on a typical desktop computer
- out-of-the-box PSS/E raw and dyr file support for available models. Once a model is developed, inputs from a dyr file can be readily supported
- always up-to-date equation documentation of implemented models

ANDES is currently under active development. To get involved,

- Follow the tutorial at <https://andes.readthedocs.io>
- Checkout the Notebook examples in the `examples` folder
- Try ANDES in Jupyter Notebook [with Binder](#)
- Download the PDF manual at [download](#)

- Report issues in the [GitHub issues page](#)
- Learn version control with the command-line `git` or [GitHub Desktop](#)
- If you are looking to develop models, read the [Modeling Cookbook](#)

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 made open source as part of the CURENT Large Scale Testbed project.

ANDES is developed and actively maintained by [Hantao Cui](#). See the GitHub repository for a full list of contributors.

## 1.2 Installation

### 1.2.1 New to Python

#### Setting Up Mambaforge

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

Step 1:

Downloaded the latest Mambaforge for your platform from <https://github.com/conda-forge/miniforge#mambaforge>. Most users will use `x86_64` (`amd64`) for Intel and AMD processors. Mac users with Apple Silicon should use `arm64` (`Apple Silicon`) for best performance.

Next, complete the Mambaforge installation on your system.

---

**Note:** Mambaforge is a drop-in replacement for conda. If you have an existing conda installation, you can replace all the following mamba commands with conda and achieve the same outcome.

---

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

---

Step 2:

Open Terminal (on Linux or macOS) or *Miniforge Prompt* (on Windows, **not cmd!!**). Make sure you are in a conda environment - you should see `(base)` prepended to the command-line prompt, such as `(base) C:\Users\username>`.

Create an environment for ANDES (recommended)

```
mamba create --name andes python=3.8
```

Activate the new environment with

```
mamba activate andes
```

---

**Note:** You will need to activate the `andes` environment every time in a new Miniforge Prompt or shell.

---

If you have completed these steps without error, you now have a working Python environment. See the commands at the top for [Getting started](#).

## 1.2.2 Extra support package

Some ANDES features require extra support packages that are not installed by default. For example, to build the documentation, one will need to install development packages. Other packages will be required for interoperability.

The extra support packages are specified in groups. The following group names are supported, with descriptions given below:

- `dev`: packages to support development such as testing and documentation
- `interop`: packages to support interoperability of ANDES and other power systems tools.

---

**Note:** Extra support packages are not supported by conda/mamba installation. One needs to install ANDES with pip.

---

To install packages in the `dev` when installing ANDES, do:

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

To install all extra packages, do:

```
pip install andes[all]
```

One can also inspect the `requirements-extra.txt` to identify the packages for manual installation.

## 1.2.3 Develop Install

The development mode installation is for users who want to modify the code and, for example, develop new models or routines. The benefit of development mode installation is that changes to source code will be reflected immediately without re-installation.

Step 1: Get ANDES source code

As a developer, you are strongly encouraged to clone the source code using `git` from either your fork or the original repository. Clone the repository with

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

---

**Note:** Replace the URL with yours to use your fork. With `git`, you can later easily update the source code and perform version control.

---

Alternatively, you can download the ANDES source code from <https://github.com/cuihantao/andes> and extract all files to the path of your choice. Although works, this method is discouraged because tracking changes and pushing back code edits will require significant manual efforts.

#### Step 2: Install dependencies

In the Mambaforge environment, use `cd` to change directory to the ANDES root folder. The folder should contain the `setup.py` file.

Install dependencies with

```
mamba install --file requirements.txt  
mamba install --file requirements-extra.txt
```

Alternatively, you can install them with `pip`:

```
pip install -r requirements.txt  
pip install -r requirements-extra.txt
```

#### Step 3: Install ANDES in the development mode using

```
python3 -m pip install -e .
```

Note the dot at the end. Pip will take care of the rest.

---

**Note:** The ANDES version number shown in `pip list` will stuck at the version that was intalled, unless ANDES is develop-installed again. It will not update automatically with `git pull`.

To check the latest version number, check the preamble by running the `andes` command or chek the output of `python -c "import andes; print(andes.__version__)"`

---

---

**Note:** ANDES updates may infrequently introduce new package requirements. If you see an `ImportError` after updating ANDES, you can manually install the missing dependencies or redo [Step 2](#).

---

---

**Note:** To install extra support packages, one can append `[NAME_OF_EXTRA]` to `pip install -e ..`. For example, `pip install -e .[interop]` will install packages to support interoperability when installing ANDES in the development, editable mode.

---

## 1.2.4 Updating ANDES

**Warning:** If ANDES has been installed in the development mode using source code, you will need to use `git` or the manual approach to update the source code. In this case, Do not proceed with the following steps, as they will install a separate site-package installation on top of the development one.

Regular ANDES updates will be pushed to both `conda-forge` and Python package index. It is recommended to use the latest version for bug fixes and new features. We also recommended you to check the [Release notes](#) before updating to stay informed of changes that might break your downstream code.

Depending you how you installed ANDES, you will use one of the following ways to upgrade.

If you installed it from mamba or conda, run

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

If you install it from PyPI (namely, through `pip`), run

```
python3 -m pip install --yes andes
```

## 1.2.5 Uninstall Multiple Copies

A common mistake new users make is to have multiple copies of ANDES installed in the same environment. This can happen when one previously installed ANDES in the development mode but later ran `conda install` or `python3 -m pip install` to install the latest version. As a result, only the most recently installed version will be accessible.

In this case, we recommend that you uninstall all version and reinstall only one copy using your preferred mode. Uninstalling all copies can be done by calling `conda remove andes` and `python3 -m pip uninstall andes`. The prompted path will indicate the copy to be removed. One may need to run the two commands for a couple of time until the package managers indicate that the `andes` package can no longer be found.

## 1.2.6 Troubleshooting

If you get an error message on Windows, reading

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

It is a path issue of your Python. In fact, Python on Windows is so broken that many people are resorting to WSL2 just for Python. Fixes can be convoluted, but the easiest one is to install ANDES in a Conda/Mambaforge environment.

## 1.3 Tutorial

ANDES can be used as a command-line tool or a library. The command-line interface (CLI) comes in handy to run studies. As a library, it can be used interactively in the IPython shell or the Jupyter Notebook. This chapter describes the most common usages.

Please see the CLI [cheatsheet](#) if you are looking for quick help.

### 1.3.1 Command line

#### Basics

ANDES is invoked from the command line using the command `andes`. Running `andes` without any input is equal to `andes -h` or `andes --help`. It prints out a preamble with version and environment information, followed by and help commands

```
| Version 1.6.0
/ \ - - | | ____ | Python 3.9.10 on Linux, 03/12/2022 10:30:44 AM
/ \ \ | ' \/ _` / -_-< |
/ \ \ \_\_||\_\_,\_\_/_/ | This program comes with ABSOLUTELY NO WARRANTY.

usage: andes [-h] [-v {1,10,20,30,40}]
              {run,plot,doc,misc,prepare,prep,selftest,st,demo} ...

positional arguments:
{run,plot,doc,misc,prepare,prep,selftest,st,demo}
    [run] run simulation routine; [plot] plot
    results; [doc] quick documentation; [misc] misc.
    functions; [prepare] prepare the numerical code;
    [selftest] run self test;

optional arguments:
-h, --help            show this help message and exit
-v {1,10,20,30,40}, --verbose {1,10,20,30,40}
                      Verbosity level in 10-DEBUG, 20-INFO, 30-WARNING,
or 40-ERROR.
```

---

**Note:** If the `andes` command is not found, it could be due to

- (1) missed steps in your installation process
  - (2) errors during installation
  - (3) forgot to activated the environment with ANDES
- 

`andes` accepts an optional argument to control verbosity level. It is done through `-v LEVEL` or `--verbose LEVEL`, where `level` is a number. Logging level by default is 20 (INFO) and can be chosen from:

- 1 (DEBUG with code location info)

- 10 (DEBUG)
- 20 (INFO)
- 30 (WARNING)
- 40 (ERROR)
- 50 (CRITICAL)

To show debugging outputs, use `andes -v 10`, followed by top-level commands. To only show warnings and errors, use `andes -v 30`.

The top-level commands are `{run, plot, doc, misc, prepare, selftest}`. Each command contains a group of subcommands, which can be looked up with `-h`. For example, use `andes run -h` to look up the subcommands for `andes run`. Frequently used commands are explained below.

---

**Note:** Some subcommands have shorthand names:

- `andes st` is equivalent to `andes selftest`
  - `andes prep` is equivalent to `andes prepare`
- 

## andes selftest

After the installation, please run `andes selftest` from the command line to test ANDES functionality. It might take a minute to run the full self-test suite. An example output looks like

```
test_docs (test_1st_system.TestCodegen) ... ok
...
... (outputs are truncated)
...
test_pflow_mpc (test_pflow_matpower.TestMATPOWER) ... ok
-----
Ran 60 tests in 10.109s
OK
```

There may be more test than what is shown above. Make sure that all tests have passed.

ANDES receives frequent updates. After each update, please run `andes st` to confirm the functionality. The command also makes sure the generated code is up to date. See [`andes prepare`](#) for more details on automatic code generation.

---

**Note:** There is a quick mode to test ANDES by skipping code generation. This should only be used when you are certain that there is no modification to models between the last code generation and now.

The quick mode is invoked by `andes st -q`.

---

## **andes prepare**

The symbolically defined models in ANDES need to be generated into numerical code for simulation. The code generation process is *automatic* the first time you use ANDES to run any case study. It takes 10 seconds to one minute to generate the code depending on your platform. When done, no code generation is needed in your future use until you modify the models.

It is also possible to generate the code manually with `andes prepare` or `andes prep`. In addition, `andes selftest` automatically calls the code generation.

---

**Note:** Generated code files are stored in Python code in `$HOME/.andes/pycode`. While being human-readable, they are not human-friendly and should only be consulted during low-level debugging.

---

The default code generation mode is known as the "quick mode". It skips the generation of *LATEX*-formatted equations, which are only useful in documentation and the interactive mode.

Option `-i` or `--incremental` can be used to speed up code generation during model development. `andes prepare -i` only generates code for models that are detected with changes since the last code generation.

**Warning:** To developers:

`andes prepare -i` needs to be called following model changes, such as equation modification and adding variables. Otherwise, due to mismatches in model and code, simulation results will not reflect the new changes, at best, or even lead to unexpected errors

ANDES supports precompiling the generated Python code using Numba. See [Numba compilation](#). Numba needs to be installed separately. Check the version of installed numba and other dependencies with `andes misc --version`.

## **andes run**

`andes run` is the entry point for power system analysis routines. The full list of options can be printed with `andes run -h`. `andes run` takes one positional argument, `filename`, along with other optional keyword arguments. `filename` is the path to cases, either relative or absolute.

- **Relative path:** `andes run kundur_full.xlsx`, e.g., uses a relative path. It works only if `kundur_full.xlsx` exists in the *working directory* of the command line.
- **Absolute path:** `andes run /Users/hcui7/kundur_full.xlsx` (on macOS) or `andes run C:/Users/hcui7/kundur_full.xlsx` (on Windows) use absolute paths to the case files. They do not depend on the command-line current directory.

---

**Note:** When working with the command line, use `cd` to change directory to the folder containing your test case. Spaces in folder and file names need to be escaped properly, so it's generally advised to *avoid spaces in file and folder names*.

---

To find out your current working directory, look for the line below the ANDES preamble that reads like

```
Working directory: "/home/hacui/repos/andes/andes/cases/kundur"
```

## Input path

ANDES allows one to specify the path to look for the case file instead of the working directory. This is done by using the `-p` or `--input-path` option. For example, if `kundur_full.xlsx` is in folder `/home/hacui/cases`, one can do

```
andes run kundur_full.xlsx -p /home/hacui/cases
```

The argument passed to `-p` or `--input-path` can also be a relative path. If you need further help understanding paths, please consult other online articles.

## Multiprocessing

ANDES takes multiple files inputs or wildcard. Multiprocessing will be triggered if more than one valid input files are passed to filename.

- Multiple files: to run the power flow for `kundur_full.xlsx` and `kundur_motor.xlsx` simultaneously, one can do

```
andes run kundur_full.xlsx kundur_motor.xlsx
```

The output will look like

```
Working directory: "/home/hacui/repos/andes/andes/cases/kundur"
-> Processing 2 jobs on 12 CPUs.
Process 0 for "kundur_full.xlsx" started.
Process 1 for "kundur_motor.xlsx" started.
Log saved to "/tmp/andes/andes-uopdutii/andes.log".
-> Multiprocessing finished in 2.4680 seconds.
```

- Wildcard: to run power flow for files with a prefix of `kundur_` and a suffix (file extension) of `.xlsx`, run

```
andes run kundur_*.xlsx
```

Case files with such name pattern, including `kundur_full.xlsx` and `kundur_motor.xlsx`, among others, will be processed.

Option `--ncpu NCPU` can be used to specify the maximum number of parallel processes. By default, all cores will be used. A small number can be specified to increase operating system responsiveness.

## Routine

Option `-r` or `-routine` is used for specifying the analysis routine, followed by the routine name. Available routine names include

- `pflow` for power flow calculation
- `tds` for time domain simulation
- `eig` for eigenvalue analysis

If `-r` is not given, the power flow calculation routine will be called. There are routine specific options that can be passed to `andes run` and are discussed next.

Each routine has a list of configuration options (called "config") to control their behaviors. Config needs to be distinguished from command-line options as not all config options are available in the command-line. Refer to [Config](#) for details.

## Power flow

---

**Note:** Examples in the following will utilize the `kundur_full.xlsx` test case. If you have cloned the ANDES repository, it can be found in `andes/cases/kundur` in the source code folder. You can also download it from [here](#).

---

To run power flow, change to the directory containing `kundur_full.xlsx`, and execute the following in the command line:

```
andes run kundur_full.xlsx
```

Alternatively, the full path to the case file is also recognizable, such as

```
andes run /home/user/andes/cases/kundur/kundur_full.xlsx
```

The power flow report will be saved to the current directory where ANDES is invoked. The report contains four sections:

- 1) system statistics,
- 2) ac bus and dc node data
- 3) ac line data, and
- 4) results of other algebraic variables and state variables.

By default, the power flow routine is configured to use full Newton Raphson method, and reactive power limits are not checked. To change these config, edit the config file by referring to `andes doc PFlow` and `andes doc PV`.

Following power flow, ANDES does not initialize dynamic models to save time. When developing dynamic models, one can enable the initialization by setting in the config file

```
[PFlow]
init_tds = 1
```

## Time-domain simulation

To run the time domain simulation (TDS) for kundur\_full.xlsx, run

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

The output looks like:

```
Parsing input file "kundur_full.xlsx"...
Input file parsed in 0.1533 seconds.
System internal structure set up in 0.0174 seconds.
-> System connectivity check results:
No islanded bus detected.
System is interconnected.
Each island has a slack bus correctly defined and enabled.

-> Power flow calculation
Sparse solver: KLU
Solution method: NR method
Numba compilation initiated with caching.
Power flow initialized in 0.1428 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 0.0014 seconds.
Report saved to "kundur_full_out.txt" in 0.0004 seconds.

-> Time Domain Simulation Summary:
Sparse Solver: KLU
Simulation time: 0.0-20.0 s.
Fixed step size: h=33.33 ms. Shrink if not converged.
Numba compilation initiated with caching.
Initialization for dynamics completed in 0.0626 seconds.
Initialization was successful.
<Toggle 1>: Line.Line_8 status changed to 0 at t=2.0 sec.
100%| #####| 100/100 [00:00<00:00, 241.53%/
→s]
Simulation completed in 0.4141 seconds.
Outputs to "kundur_full_out.lst" and "kundur_full_out.npz".
Outputs written in 0.0171 seconds.
-> Single process finished in 0.8890 seconds.
```

The output contains the key information for the simulation, such as solver name and step size. It prints out the disturbance information, the trip of Line Line\_8 at time t=2.0 sec.

There are a few places needing to be noted:

1. Make sure the power flow calculation is successful. Otherwise, there is no good starting point for dynamic simulation.
2. Make sure no suspect initialization error is found. Otherwise, the system will not be at steady state even before disturbances.

TDS writes two output files: a variable list file `kundur_full_out.lst`, and a compressed NumPy data file `kundur_full_out.npz`:

- List file: it is a plain-text file with three columns: variable indices, variable name in plain text, and variable name in the *LATEX* format. The variable indices are needed to plot the needed variable.
- Data file: it is a zipped NumPy binary file. Although not directly editable, it can be used for plotting or can be converted to a CSV file. See the subsection on [andes plot](#).

There are TDS-specific options that can be passed to `andes run`:

- `--tf TF`: the final time of the simulation. TF should be a number in seconds. By default, it is set to 20.0.
- `--addfile ADDFILE`: specify an additional data file. This is currently used to supply PSS/E dyr file in addition to a raw file.
- `--flat`: turn on "flat run" mode to ignore all disturbances. The simulation will be performed up to the end time.
- `--no-pbar`: turn off progress bar.
- `--from-csv FROM_CSV`: use data from a CSV file to perform mock simulation. The CSV file should be in the format of `andes plot --to-csv`.

## Disable output

Output to files can be disabled with `--no-output` or `-n`. It is useful when computation is needed but results can be discarded. It is also useful when results are processed in memory, combined with the `--shell` option discussed next.

## IPython shell

The ANDES CLI will exit to the system shell when finished running. It is sometimes useful to script in Python to quickly process the simulation results in memory, such as plotting. ANDES can exit to the IPython shell with `--shell` or `-s`. For example:

```
andes run kundur_full.xlsx -r tds -s -n
```

Note the `-n` is optional to disable file output. The terminal output will look like

```
<Toggle 1>: Line.Line_8 status changed to 0 at t=2.0 sec.  
100%|#####| 100/100 [00:00<00:00, 246.07%/  
→s]  
Simulation completed in 0.4064 seconds.
```

(continues on next page)

(continued from previous page)

```
Outputs to "kundur_full_out.lst" and "kundur_full_out.npz".
Outputs written in 0.0171 seconds.
-> Single process finished in 0.8796 seconds.
IPython: Access System object in variable `system`.
Python 3.9.10 | packaged by conda-forge | (main, Feb 1 2022, 21:24:11)
Type 'copyright', 'credits' or 'license' for more information
IPython 8.1.1 -- An enhanced Interactive Python. Type '?' for help.
```

In [1]:

A prompt will appear following In [1]: to indicate an IPython shell. If the test case file is parsed without error, the system object will be stored in variable `system`, i.e.

```
In [1]: system
Out[1]: <andes.system.System at 0x7fc1cd992790>
```

Python commands can be executed thereafter. To exit, type `exit` and press enter.

## Format converter

ANDES uses the Excel format to store power system data in the ANDES semantics. In addition, multiple input formats are recognized and can be converted to the ANDES `xlsx` format. Converting data into the ANDES has pros and cons:

- Pros: - The data can be readily edited with an Excel-like tool - Data for models unique to ANDES can be readily added to the `xlsx` file
- Cons: - Conversion from ANDES `xlsx` back to the original format is not supported

**Note:** It is recommended to stay with the original data format to maximize compatibility when no ANDES-specific models are used.

Format conversion is done through `--convert FORMAT` or `-c FORMAT`, where `FORMAT` is the output format. For now, the following formats are supported:

- `xlsx`: an Excel spread sheet format with ANDES-specific data. It is not compatible with `xlsx` with data from other tools such as [Pandapower](#).
- `json`: a JSON plain-text file with ANDES-specific data. Likewise, it is unlikely to be compatible with JSON from other power system tools. JSON is much faster to parse than `xlsx` but not as friendly to edit.

To convert `kundur_full.xlsx`, for example, to the `json` format, run

```
andes run kundur_full.xlsx --convert json
```

The output messages will look like

```
Parsing input file "kundur_full.xlsx"...
Input file parsed in 0.1576 seconds.
System internal structure set up in 0.0175 seconds.
JSON file written to "kundur_full.json"
Format conversion completed in 0.0053 seconds.
-> Single process finished in 0.2582 seconds.
```

Note that --convert will only create sheets for existing models.

The converter works with any input formats that are currently supported. These include:

- .m: MATPOWER case file
- .raw and .dyr: PSS/E raw and dyr files
- .xlsx: Excel spreadsheet file with ANDES data
- .json: JSON plain-text file with ANDES data

## PSS/E inputs

To work with PSS/E input files (.raw and .dyr), one need to provide the raw file through `casefile` and pass the dyr file through `--addfile`. For example, in `andes/cases/kundur`, one can run the power flow using

```
andes run kundur.raw
```

and run a no-disturbance time-domain simulation using

```
andes run kundur.raw --addfile kundur_full.dyr -r tds
```

---

**Note:** If one wants to modify the parameters of models that are supported by both PSS/E and ANDES, one can directly edit those dynamic parameters in the `.raw` and `.dyr` files to maintain interoperability with other tools.

---

To create add a disturbance, there are two options. The recommended option is to convert the PSS/E data into an ANDES xlsx file, edit it and run (see the previous subsection). The alternative approach is documented in [Creating disturbances](#).

## Profiling

Profiling is useful for analyzing the computation time and code efficiency. Option `--profile` enables the profiling of ANDES execution. The profiling output will be written in two files in the current folder, one ending with `_prof.txt` and the other one with `_prof.prof`.

The text file can be opened with a text editor, and the `.prof` file can be visualized with `snakeviz`, which can be installed with `pip install snakeviz`.

If the output is disabled, profiling results will be printed to `stdio`.

## andes plot

`andes plot` is the command-line tool for plotting. It currently supports time-domain simulation data. Three positional arguments are required, and a dozen of optional arguments are supported.

positional arguments:

Argument	Description
filename	simulation output file name, which should end with <code>out</code> . File extension can be omitted.
x	the X-axis variable index, typically 0 for Time
y	Y-axis variable indices. Space-separated indices or a colon-separated range is accepted

For the list of optional arguments, see the output of `andes plot -h`.

To plot the generator speed variable `omega` of `GENROU_1 omega` `GENROU 1` versus time, one way is to supply the variable indices found in the `.lst` output file. The index of the variable `omega GENROU 1` is found to be 5, and Time is found to be 0, so the plot command should be

```
andes plot kundur_full_out.lst 0 5
```

where `kundur_full_out.lst` is list file name, 0 is the index of Time for the x-axis, and 5 is the index of `omega GENROU 1`. Note that for the the file name, either `kundur_full_out.lst` or `kundur_full_out.npz` works as the program will automatically extract the file name.

The y-axis variable indices can also be specified as a Python range. For example, `andes plot kundur_full_out.npz 0 2:21:6` will plot the variables with indices 2, 8, 14 and 20.

It can become tedious to look up the indices of variables in the `.lst` file. `andes plot` supports `--xargs` or `-a` for searching for variable indices and passing them as arguments to `andes plot`. See Examples - "Using CLI from Notebook".

## LaTeX rendering

`andes plot` will attempt to render with  $\text{\textit{LATEX}}$  if `dvipng` program is in the search path. Figures rendered by  $\text{\textit{LATEX}}$  has publication-quality aesthetics for symbols but takes considerably longer time. In case  $\text{\textit{LATEX}}$  is available but fails (frequently happens on Windows), the option `-d` can be used to disable  $\text{\textit{LATEX}}$  rendering.

## andes doc

`andes doc` is a handy tool to look up model, routine and config documentation. Model documentation include the descriptions of parameters, variables, and configs. A pretty-print version is available online in [Model reference](#).

The basic usage of `andes doc` is to provide a model name or a routine name as the positional argument. For a model, it will print out model parameters, variables, and equations to the stdio. For a routine, it will print out fields in the Config file.

---

**Note:** For full model documentation, visit [Model reference](#).

---

For example, to check the parameters for model `Toggle`, run

Name	Description	Default	Unit	Type	
<a href="#">Properties</a>					
<a href="#">--</a>					
u	connection status	1	bool	NumParam	
name	device name			DataParam	
model	Model or Group of the device			DataParam	
<a href="#">mandatory</a>					
	to control				
dev	idx of the device to control			IdxParam	
<a href="#">mandatory</a>					
t	switch time for connection	-1		TimerParam	
<a href="#">mandatory</a>					
	status				

To list all supported models, run

Group	Models
<a href="#">Supported Groups and Models</a>	
ACLine	Line
ACTopology	Bus
Collection	Area
DCLink	Ground, R, L, C, RCp, RCs, RLs, RLCs, RLcp
DCTopology	Node
Exciter	EXDC2
Experimental	PI2
FreqMeasurement	BusFreq, BusROCOF

(continues on next page)

(continued from previous page)

StaticACDC	VSCShunt
StaticGen	PV, Slack
StaticLoad	PQ
StaticShunt	Shunt
SynGen	GENCLS, GENROU
TimedEvent	Toggle, Fault
TurbineGov	TG2, TGOV1

To view the Config fields for a routine, run

Option	Value	Info	Acceptable
<code>→values</code>			
<code>→---</code>			
sparselib	klu	linear sparse solver name	('klu', 'umfpack')
tol	0.000	convergence tolerance	float
t0	0	simulation starting time	>=0
tf	20	simulation ending time	>t0
fixt	0	use fixed step size (1) or variable	(0, 1)
	(0)		
shrinkt	1	shrink step size for fixed method if	(0, 1)
	not converged		
tstep	0.010	the initial step step size	float
max_iter	15	maximum number of iterations	>=10

## andes misc

`andes misc` contains miscellaneous functions, such as configuration and output cleaning.

### Configuration

ANDES uses a configuration file to set runtime configs for the system routines, and models. `andes misc --save-config` saves all configs to a file. By default, it saves to `$HOME/.andes/andes.conf` file, where `$HOME` is the path to your home directory.

With `andes misc --edit-config`, you can edit ANDES configuration handy. The command will automatically save the configuration to the default location if not exist. The shorter version `--edit` can be used instead as Python matches it with `--edit-config`.

You can pass an editor name to `--edit`, such as `--edit vim`. If the editor name is not provided, it will use the following defaults: - Microsoft Windows: notepad. - GNU/Linux: the `$EDITOR` environment variable, or `vim` if not exist.

For macOS users, the default is `vim`. If not familiar with `vim`, you can use `nano` with `--edit nano` or `TextEdit` with `--edit "open -a TextEdit"`.

## Cleanup

```
andes misc -C, --clean
```

Option to remove any generated files. Removes files with any of the following suffix: `_out.txt` (power flow report), `_out.npy` (time domain data), `_out.lst` (time domain variable list), and `_eig.txt` (eigenvalue report).

## Version

Check the version of ANDES and the core packages it uses, use

```
andes misc --version
```

Please include the output in your bug report.

## 1.3.2 Scripting

This section is a tutorial for using ANDES in an interactive/scripting environment. All scripting shells are supported, including Python shell, IPython, Jupyter Notebook, and Jupyter Lab. The examples below use Jupyter Notebook.

### Jupyter Notebook

Jupyter Notebook is a convenient tool to run Python code and present results. Jupyter Notebook can be installed with

```
conda install jupyter notebook
```

After the installation, change the directory to the folder where you wish to store notebooks, then start the notebook with

```
jupyter notebook
```

A browser window should open automatically with the notebook browser loaded. To create a new notebook, use the "New" button near the upper-right corner.

---

**Note:** In the following, code blocks starting with `>>>` are Python code and should be executed inside Python, IPython, or Jupyter Notebook. Python code should not be entered into Anaconda Prompt or Linux shell.

---

## Import

Like other Python libraries, ANDES needs to be imported into an interactive scripting Python environment.

```
>>> import andes
>>> andes.config_logger()
```

## Verbosity

If you are debugging ANDES, you can enable debug messages with

```
>>> andes.config_logger(stream_level=10)
```

or simply

```
>>> andes.config_logger(10)
```

The `stream_level` uses the same verbosity levels as for the command line. If not explicitly enabled, the default level 20 (INFO) will apply.

To set a new logging level for the current session, call `config_logger` with the desired new levels.

## Making a System

Before running studies, an `andes.system.System` object needs to be created to hold the system data. The `System` object can be created by passing the path to the case file to the entry-point function.

There are multiple ways to create the `System` object. Among them, `andes.main.run` is the most convenient way. For example, to run the file `kundur_full.xlsx` in the same directory as the notebook, use

```
>>> ss = andes.run('kundur_full.xlsx')
```

This function will parse the input file, run the power flow, and return the system as an object. Outputs will look like

```
Parsing input file </Users/user/notebooks/kundur/kundur_full.xlsx>
Input file kundur_full.xlsx parsed in 0.4172 second.
-> Power flow calculation with Newton Raphson method:
0: |F(x)| = 14.9283
1: |F(x)| = 3.60859
2: |F(x)| = 0.170093
3: |F(x)| = 0.00203827
4: |F(x)| = 3.76414e-07
Converged in 5 iterations in 0.0222 second.
Report saved to </Users/user/notebooks/kundur_full_out.txt> in 0.0015 second.
-> Single process finished in 0.4677 second.
```

In this example, `ss` is an instance of `andes.System`. It contains member attributes for models, routines, and numerical DAE.

The naming convention for the System attributes is as follows

- Model attributes share the same name as class names. For example, `ss.Bus` is the Bus instance, and `ss.GENROU` is the GENROU instance.
- Routine attributes share the same name as class names. For example, `ss.PFlow` and `ss.TDS` are the routine instances.
- The numerical DAE instance is in lower case `ss.dae`.

To work with PSS/E inputs, refer to [Examples](#) - "Working with Data".

---

**Note:** `andes.main.run` can accept multiple input files for multiprocessing. They can be passed as a list of strings to the first positional argument.

---

## Passing options

`andes.run()` can accept options that are available to the command-line `andes run`. Options need to be passed as keyword arguments to `andes.run()` in addition to the positional argument for the test case. For example, setting `no_output` to `True` will disable all file outputs. When scripting, one can do

```
>>> ss = andes.run('kundur_full.xlsx', no_output=True)
```

which is equivalent to the following shell command:

```
andes run kundur_full.xlsx --no-output
```

Please note that the dash between `no` and `output` needs to be replaced with an underscore for scripting. This is the convention in Python's argument parser.

Another example is to specify a folder for output files. By default, outputs will be saved to the folder where Python is run (or where the notebook is run). In case you need to organize outputs, a path prefix can be passed to `andes.run()` through `output_path`:

```
>>> ss = andes.run('kundur_full.xlsx', output_path='outputs/')
```

which will put outputs into folder `outputs` relative to the current path. You can also supply an absolute path to `output_path`.

The next example is to specify the simulation time for a time-domain simulation. There are multiple ways to implement it (see [Examples](#)), and one way is to pass the end time (in sec) through argument `tf` and set the routine to `tds`:

```
>>> ss = andes.run('kundur_full.xlsx', routine='tds', tf=5)
```

which will set the simulation time to 5 seconds.

---

**Note:** While `andes run` accepts single-letter alias for the option, such as `andes run -n` for `andes run --no-output`, `andes.run()` can only work with the full option name (with hyphen replaced by

---

underscore)

---

## Load Only

In many workflows, one will simulate many scenarios with largely identical system data. A base case can be loaded and modified to create scenarios in memory. See Example "Working with Data" for details

## Inspecting Parameter

### DataFrame

Parameters for the loaded system can be readily inspected in Jupyter Notebook using Pandas.

Parameters for a model instance can be retrieved in a DataFrame using the `as_df()` method on the model instance. For example, to view the parameters of Bus, use

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

A table will be printed with the columns being parameters and the rows being Bus devices/instances. For a system that has been setup, parameters have been converted to per unit values in the system base specified by `ss.config.mva`. The per-unit values in the system base will be used in computation as all computation in ANDES uses system-base per-unit data.

To view the original input values, use the `as_df(vin=True)` method. For example, to view the system-base per unit value of GENROU, use

```
>>> ss.GENROU.as_df(vin=True)
```

Parameter in the table is the same as that in the input file without any conversion. Some input data, by convention, are given as per unit in the device base; see [Per Unit System](#) for details.

Note that `andes.core.modeldata.ModelData.as_df()` returns a *view*. Modifying the returned dataframe *will not* affect the original data used for simulation. To modify the data, see Example "Working with Data".

## Running Studies

Three routines are currently supported: PFlow, TDS and EIG. Each routine provides a `run()` method to execute. The System instance contains member attributes having the same names. For example, to run the time-domain simulation for `ss`, use

```
>>> ss.TDS.run()
```

To change configuration for routines, one can set the attribute before calling run. For example, to change the end time to 5 sec, one can do

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

Note that not all config changes are respected. Some config values are used while creating the routine instance. For config changes that does not necessarily have to be done on-the-fly, it is recommended to edit the config file.

## Checking Exit Code

`andes.System` contains field `exit_code` for checking if error occurred in run time. A normal completion without error should always have `exit_code == 0`. One should read output messages carefully and check the exit code, which is particularly useful for batch simulations.

Error may occur in any phase - data parsing, power flow, or simulation. To diagnose, split the simulation steps and check the outputs from each one.

## Plotting TDS Results

TDS comes with a plotting utility for scripting usage. After running the simulation, a `plotter` attributed will be created for `TDS`. To use the plotter, provide the attribute instance of the variable to plot. For example, to plot all the generator speed, use

```
>>> ss.TDS.plotter.plot(ss.GENROU.omega)
```

---

**Note:** If you see the error

AttributeError: 'NoneType' object has no attribute 'plot'

You will need to manually load plotter with

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

---

Optional indices is accepted to choose the specific elements to plot. It can be passed as a tuple through the `a` argument

```
>>> ss.TDS.plotter.plot(ss.GENROU.omega, a=(0, ))
```

In the above example, the speed of the "zero-th" generator will be plotted.

## Scaling

A lambda function can be passed to argument `ycalc` to scale the values. This is useful to convert a per-unit variable to nominal. For example, to plot generator speed in Hertz, use

```
>>> ss.TDS.plotter.plot(ss.GENROU.omega, a=(0, ),  
                      ycalc=lambda x: 60*x,  
                      )
```

## Formatting

A few formatting arguments are supported:

- `grid` = `True` to turn on grid display
- `greyscale` = `True` to switch to greyscale
- `ylabel` takes a string for the y-axis label

## Extracting Data

One can extract data from ANDES for custom plotting. Variable names can be extracted from the following fields of `ss.dae`:

Un-formatted names (non-LaTeX):

- `x_name`: state variable names
- `y_name`: algebraic variable names
- `xy_name`: state variable names followed by algebraic ones

LaTeX-formatted names:

- `x_tex_name`: state variable names
- `y_tex_name`: algebraic variable names
- `xy_tex_name`: state variable names followed by algebraic ones

These lists only contain the variable names used in the current analysis routine. If you only ran power flow, `ss.dae.y_name` will only contain the power flow algebraic variables, and `ss.dae.x_name` will likely be empty. After initializing time-domain simulation, these lists will be extended to include all variables used by TDS.

In case you want to extract the discontinuous flags from TDS, you can set `store_z` to 1 in the config file under section `[TDS]`. When enabled, discontinuous flag names will be populated at

- `ss.dae.z_name`: discontinuous flag names
- `ss.dae.z_tex_name`: LaTeX-formatted discontinuous flag names

If not enabled, both lists will be empty.

## Power flow solutions

The full power flow solutions are stored at `ss.dae.xy` after running power flow (and before initializing dynamic models). You can extract values from `ss.dae.xy`, which corresponds to the names in `ss.dae.xy_name` or `ss.dae.xy_tex_name`.

If you want to extract variables from a particular model, for example, bus voltages, you can directly access the `v` field of that variable

```
>>> import numpy as np  
>>> voltages = np.array(ss.Bus.v.v)
```

which stores a **copy** of the bus voltage values. Note that the first `v` is the voltage variable of `Bus`, and the second `v` stands for *value*. It is important to make a copy by using `np.array()` to avoid accidental changes to the solutions.

If you want to extract bus voltage phase angles, do

```
>>> angle = np.array(ss.Bus.a.v)
```

where `a` is the field name for voltage angle.

To find out names of variables in a model, use command `andes doc` or refer to *Model reference*.

## Time-domain data

Time-domain simulation data will be ready when simulation completes. It is stored in `ss.dae.ts`, which has the following fields:

- `txyz`: a two-dimensional array. The first column is time stamps, and the following are variables. Each row contains all variables for that time step.
- `t`: all time stamps.
- `x`: all state variables (one column per variable).
- `y`: all algebraic variables (one column per variable).
- `z`: all discontinuous flags (if enabled, one column per flag).

If you want the output in pandas DataFrame, call

```
ss.dae.ts.unpack(df=True)
```

Dataframes are stored in the following fields of `ss.dae.ts`:

- `df`: dataframe for states and algebraic variables
- `df_z`: dataframe for discontinuous flags (if enabled)

For both dataframes, time is the index column, and each column correspond to one variable.

---

**Note:** Looking to extract data for a single variable? See [Examples](#) - "Working with Data".

---

## Pretty Print of Equations

Each ANDES models offers pretty print of *LATEX*-formatted equations in the jupyter notebook environment. To use this feature, symbolic equations need to be generated in the current session using

```
import andes ss = andes.System() ss.prepare()
```

Or, more concisely, one can do

```
import andes ss = andes.prepare()
```

This process may take a few minutes to complete. To save time, you can selectively generate it only for interested models. For example, to generate for the classical generator model GENCLS, do

```
import andes ss = andes.System() ss.GENROU.prepare()
```

Once done, equations can be viewed by accessing `ss.<ModelName>.syms.<PrintName>`, where `<ModelName>` is the model name, and `<PrintName>` is the equation or Jacobian name.

---

**Note:** Pretty print only works for the particular `System` instance whose `prepare()` method is called. In the above example, pretty print only works for `ss` after calling `prepare()`.

---

Supported equation names include the following:

- `xy`: variables in the order of *State*, *ExtState*, *Algeb* and *ExtAlgeb*
- `f`: the **right-hand side of** differential equations  $\mathbf{M}\dot{\mathbf{x}} = \mathbf{f}$
- `g`: implicit algebraic equations  $0 = \mathbf{g}$
- `df`: derivatives of `f` over all variables `xy`
- `dg`: derivatives of `g` over all variables `xy`
- `s`: the value equations for *ConstService*

For example, to print the algebraic equations of model GENCLS, one can use `ss.GENCLS.syms.g`.

## Finding Help

### docstring

To find out how a Python class, method, or function should be used, use the built-in `help()` function. This will print out the docstring of the class/method/function. For example, to check how the `get` method of `GENROU` should be called, do

```
help(ss.GENROU.get)
```

In Jupyter notebook, this can be simplified into `?ss.GENROU.get` or `ss.GENROU.get?`.

Please report issues if you find missing docstring.

### Model docs

Model docs can be shown by printing the return of `doc()`. For example, to check the docs of `GENCLS`, do

```
print(ss.GENCLS.doc())
```

It is the same as calling `andes doc GENCLS` from the command line. Likewise, a pretty-print version is available online in [Model reference](#).

### 1.3.3 Cheatsheet

A cheatsheet is available for quick lookup of supported commands.

View the PDF version at

<https://www.cheatography.com//cuihantao/cheat-sheets/andes-for-power-system-simulation/pdf/>

### 1.3.4 Documentation

The documentation you are viewing can be made locally in a variety of formats. To make HTML documentation, change directory to `docs`, and do

```
make html
```

After a few minutes, HTML documentation will be saved to `docs/build/html` with the index page being `index.html`. You can use Python to serve it locally. In the folder `docs/build/html`, run

```
python -m http.server
```

A list of supported formats is as follows. Note that some format require additional compiler or library

```

html      to make standalone HTML files
dirhtml   to make HTML files named index.html in directories
singlehtml to make a single large HTML file
pickle    to make pickle files
json      to make JSON files
htmlhelp   to make HTML files and an HTML help project
qthelp    to make HTML files and a qthelp project
devhelp   to make HTML files and a Devhelp project
epub     to make an epub
latex    to make LaTeX files, you can set PAPER=a4 or PAPER=letter
latexpdf  to make LaTeX and PDF files (default pdflatex)
latexpdfja to make LaTeX files and run them through platex/dvipdfmx
text     to make text files
man      to make manual pages
texinfo   to make Texinfo files
info      to make Texinfo files and run them through makeinfo
gettext   to make PO message catalogs
changes   to make an overview of all changed/added/deprecated items
xml      to make Docutils-native XML files
pseudoxml to make pseudoxml-XML files for display purposes
linkcheck to check all external links for integrity
doctest   to run all doctests embedded in the documentation (if enabled)
coverage  to run coverage check of the documentation (if enabled)

```

## 1.4 Config

ANDES uses a config file to store the various options for routines and models. As discussed in [andes misc](#), the config file is at `$HOME/.andes/andes.rc`. The reference for the config fields can be found in [Config reference](#).

### 1.4.1 Format

The ANDES config file uses the format provided by Python module `configparser`. The syntax is like the following:

```

[System]
freq = 60
mva = 100
...
[PFlow]
sparselib = klu
linsolve = 0
tol = 1e-6
...
[TGOV1]
allow_adjust = 1

```

(continues on next page)

(continued from previous page)

```
adjust_lower = 0
adjust_upper = 1
```

In the above, System, PFlow and TGOV1 are two sections. freq = 60, for example, is a pair of option and value in the [System] section. Note the space before and after the equal sign.

The meaning of the fields in each section can be found in [Config reference](#), which contains the default values and acceptable values for each option. The values for config fields can be a string or a number. Fields with acceptable values being (0, 1) can only accept 0 or 1 to indicate true or false. Non-binary values for such options will cause unexpected errors in the program.

## Limits in models

All models have three config options:

- allow\_adjust: allow limits of limiters in this model to be adjusted if the inputs, at steady state, is out of the limits. allow\_adjust = 0 is the global off-switch for this model.
- adjust\_lower: allow reducing the lower limit to the input value, if the input at steady-state is below the lower limit. This is disabled by default.
- adjust\_upper: allow increasing the upper limit to the steady-state input. This is enabled by default.

Note that setting allow\_adjust = 0 is equal to setting adjust\_lower = 0 and adjust\_upper = 0, but the former saves some time for function calls.

The limit adjustment feature is to alleviate issues caused by the model parameters. Commercial tools have a more sophisticated mechanism for autocorrection, and the limit adjustment is part of it. However, if you see an limit adjustment warning or even initialization error, it is important not to rely on autocorrection but fix the data by yourself. Autocorrected data can yield some results but issues can remain hidden.

### 1.4.2 Adjusting on the fly

#### CLI

One can adjust config on the fly *in command line* without modifying or even storing the config file. This is useful when the config change is one-time or ANDES CLI is stored in a read-only container. The config update is done by andes run --config-option SECTION.OPTION=VALUE, where SECTION is the section name, OPTION is the option name, and VALUE is the new value. *No space is allowed around . and =.*

For example, to solve kundur\_full.json with reactive power limit enforced, one can do:

```
andes run kundur_full.json -O PV.pv2pq=1
```

where -O is the shorthand command for --config-option, and the enabled pv2pq will allow PV to be converted to PQ once reactive power limit is hit.

Multiple config updates can be passed simultaneously, separated by *space*. For example, to enable reactive power limit and switch the power flow solver to UMFPACK, do:

```
andes run kundur_full.json -O PV.pv2pq=1 PFlow.sparselib=umfpack
```

## Scripting

To adjust config on the fly when scripting, there are two cases:

- Update the config when creating a new System object
- Update the config for an existing System object

To update the config when creating a new System object, one can pass a list of strings to `config_option` for `andes.main.run`:

```
>>> ss = andes.run("kundur_full.json", config_option=["PV.pv2pq=1"])
```

which directly calls the backend API for the CLI. To update multiple configs, one can do

```
>>> ss = andes.run("kundur_full.json",
                    config_option=["PV.pv2pq=1", "PFlow.sparselib=umfpack"])
```

When the System object gets created, the config values will be distributed to member attributes of the System object. Therefore, the config for a System object `ss` is stored in `ss.config`, and the config for the power flow routine is stored in `ss.PFlow.config`.

To update the config for an existing system, one can directly access the `config` attribute and set the new value. To set a new simulation end time, one can overwrite the `ss.TDS.config.tf` field, such as:

```
# load system and run power flow
>>> ss = andes.run("kundur_full.json")
>>> ss.TDS.config.tf = 5.0
>>> ss.TDS.run()
```

**Warning:** Not all config options can be updated on the fly. Those config that are used for constructing the system object can only be updated when creating a new System object.

### 1.4.3 Config reference

## System

Option	Value	Info	Accepted values
freq	60	base frequency [Hz]	float
mva	100	system base MVA	float
ipadd	1	use spmatrix.ipadd if available	(0, 1)
seed	None	seed (or None) for random number generator	int or None
diag_eps	0.000	small value for Jacobian diagonals	
warn_limits	1	warn variables initialized at limits	(0, 1)
warn_abnormal	1	warn initialization out of normal values	(0, 1)
dime_enabled	0		
dime_name	andes		
dime_address	ipc:///tmp/dime		
numba	0	use numba for JIT compilation	(0, 1)
numba_parallel	0	enable parallel for numba.jit	(0, 1)
numba_nopython	1	nopython mode for numba	(0, 1)
yapf_pycode	0	format generated code with yapf	(0, 1)
save_stats	0	store statistics of function calls	(0, 1)
np_divide	warn	treatment for division by zero	{'ignore', 'call', 'raise', 'log', 'print', 'warn'}
np_invalid	warn	treatment for invalid floating-point ops.	{'ignore', 'call', 'raise', 'log', 'print', 'warn'}

## PFlow

Option	Value	Info	Accepted values
sparselib	klu	linear sparse solver name	('klu', 'umfpack', 'spsolve', 'cupy')
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')
check_conn	1	check connectivity before power flow	(0, 1)
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)

**TDS**

Option	Value	Info	Accepted values
sparselib	klu	linear sparse solver name	('klu', 'umfpack', 'spssolve', 'cupy')
linsolve	0	solve symbolic factorization each step (enable when KLU segfaults)	(0, 1)
method	trapezoid	DAE solution method	('trapezoid', 'backeuler')
tol	0.000	convergence tolerance	float
t0	0	simulation starting time	>=0
tf	20	simulation ending time	>t0
fixt	1	use fixed step size (1) or variable (0)	(0, 1)
shrinkt	1	shrink step size for fixed method if not converged	(0, 1)
honest	0	honest Newton method that updates Jac at each step	(0, 1)
tstep	0.033	integration step size	float
max_iter	15	maximum number of iterations	>=10
refresh_event	0	refresh events at each step	(0, 1)
test_init	1	test if initialization passes	(0, 1)
check_conn	1	re-check connectivity after event	(0, 1)
criteria	1	use criteria to stop simulation if unstable	(0, 1)
ddelta_limit	180	delta diff. limit to be considered unstable, in degree	
g_scale	1	scale algebraic residuals with time step size	positive
reset_tiny	1	reset tiny residuals to zero to avoid chattering	(0, 1)
qrt	0	quasi-real-time stepping	(0, 1)
kqrt	1	quasi-real-time scaling factor; kqrt > 1 means slowing down	positive
store_z	0	store limiter status in TDS output	(0, 1)
store_f	0	store RHS of diff. equations	(0, 1)
store_h	0	store RHS of external diff. equations	(0, 1)
store_i	0	store RHS of external algeb. equations	(0, 1)
limit_store	0	limit in-memory timeseries storage	(0, 1)
max_store	900	maximum steps of data stored in memory before offloading	positive integer
save_every	1	save one step to memory every N simulation steps	integer
save_mode	auto	automatically or manually save output data when done	('auto', 'manual')
no_tqdm	0	disable tqdm progressbar and outputs	(0, 1)
chatter_iter	4	minimum iterations to detect chattering	int>=4

**EIG**

Option	Value	Info	Accepted values
sparselib	klu	linear sparse solver name	('klu', 'umfpack', 'spssolve', 'cupy')
lin-solve	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	

## 1.5 Input formats

ANDES currently supports the following input formats:

- .xlsx: Excel spreadsheet file with ANDES data
- .raw and .dyr: PSS/E RAW and DYR formats
- .m: MATPOWER format
- .json: JSON plain-text format with ANDES data

### 1.5.1 ANDES xlsx

The ANDES xlsx format allows one to use Excel for convenient viewing and editing. If you do not use Excel, there are alternatives such as the free and open-source [LibreOffice](#).

#### Format definition

The ANDES xlsx format contains multiple workbooks (also known as "sheets") shown as tabs at the bottom. The name of a workbook is a *model* name, and each workbook contains the parameters of all *devices* that are *instances* of the model.

In each sheet, the first row contains the names of parameters of the model. Starting from the second row, each row corresponds to a *device instance* with the parameters in the corresponding columns. An example of the Bus sheet is shown in the following screenshot.

The screenshot shows a Microsoft Excel window with the title 'kundur\_fault'. The 'Bus' sheet is selected. The data starts with a header row (row 1) containing column labels: uid, idx, u, name, Vn, vmax, vmin, v0, a0, xcoord, ycoord, area, zone, owner. Rows 2 through 11 contain data for 10 different bus instances. Row 12 is empty. Row 13 contains the text 'All Bus Instances'. Row 14 contains the text 'Models (workbooks)'. The bottom navigation bar shows tabs for Bus, PQ, PV, Slack, Shunt, Line, Area, GENROU, TGOV1, EXDC2, and a '+' button. The zoom level is set to 100%.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	uid	idx	u	name	Vn	vmax	vmin	v0	a0	xcoord	ycoord	area	zone	owner			
2	0	1	1	1	20	1.1	0.9	1	0.570255	0	0	1	1	1			
3	1	2	1	2	20	1.1	0.9	0.99761	0.368746	0	0	1	1	1			
4	2	3	1	12	20	1.1	0.9	0.96263	0.185317	0	0	2	1	1			
5	3	4	1	11	20	1.1	0.9	0.81691	0.462359	0	0	2	1	1			
6	4	5	1	101	230	1.1	0.9	0.97928	0.480203	0	0	1	1	1			
7	5	6	1	102	230	1.1	0.9	0.95796	0.283887	0	0	1	1	1			
8	6	7	1	3	230	1.1	0.9	0.9362	0.126901	0	0	1	1	1			
9	7	8	1	13	230	1.1	0.9	0.87904	-0.08059	0	0	2	1	1			
10	8	9	1	112	230	1.1	0.9	0.89054	0.093618	0	0	2	1	1			
11	9	10	1	111	230	1.1	0.9	0.82958	0.336601	0	0	2	1	1			

## Common parameters

A few columns are used across all models. That includes `uid`, `idx`, `name` and `u`:

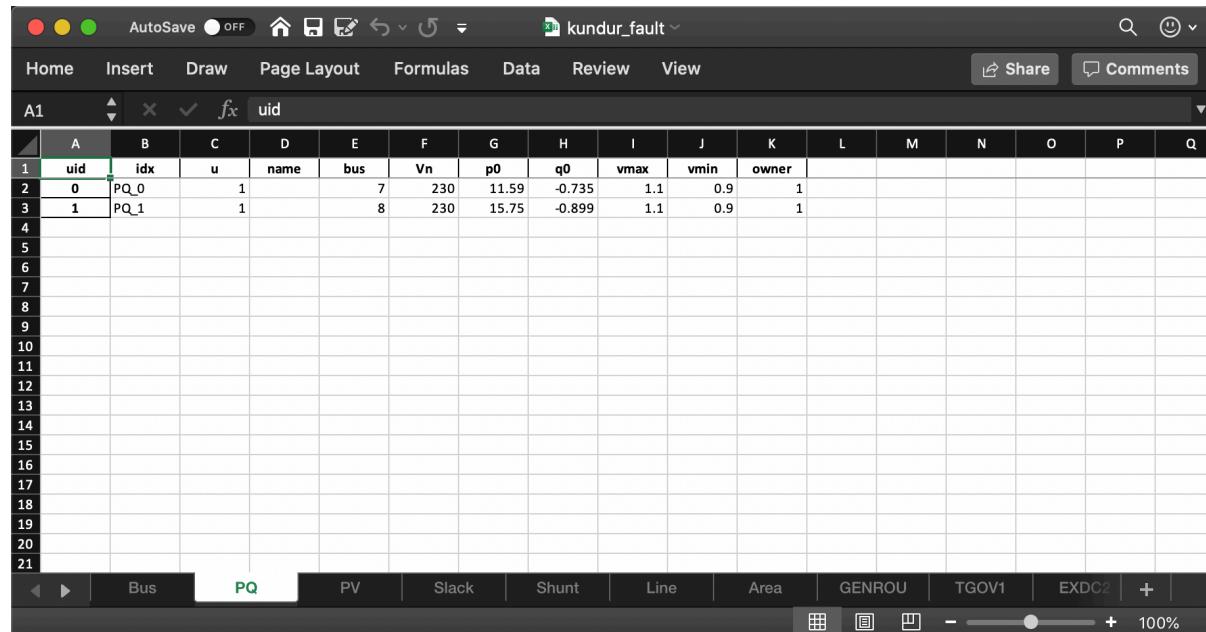
- `uid` is an unique index that is generated and used *internally*. This column can be left empty when the sheet is being created manually. Exporting systems to `xlsx` with `--convert` (see *Format converter*) will have the `uid` overwritten.
- `idx` is the *unique index to identify a device* of the model. An unique `idx` should be provided explicitly for each instance for best consistency. Accepted types for `idx` include numbers and strings without spaces.

**Warning:** ANDES will check the uniqueness of `idx` and assign new ones when duplication is detected. Duplicate `idx` indicates data inconsistency and will likely cause simulations to fail.

- `u` is the connectivity status of the instance. Accepted values are 0 for disconnected (turned off) and 1 for connected (turned on). Disconnected devices will still have the variables assigned in ANDES but will not interact with the simulation. Unexpected behaviors may occur if numerical values other than 0 and 1 are assigned, as `u` is often used as a multiplier in equations.
- `name` is the name for the device instance. It is used for display purposes and can be left empty.

## Connecting devices

Most importantly, `idx` is the *unique* index for referencing a device, so that it can be properly connected by supported devices. In a system, a PQ (constant PQ load) device needs to connect to a Bus device to inject power. That is, the PQ device needs to indicate the Bus device to which it is connected. Such connection is done in the PQ sheet by setting the `bus` parameter to the `idx` of the connected bus.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	uid	idx	u	name	bus	Vn	p0	q0	vmax	vmin	owner						
2	0	PQ_0		1		7	230	11.59	-0.735	1.1	0.9	1					
3	1	PQ_1		1		8	230	15.75	-0.899	1.1	0.9	1					
4																	
5																	
6																	
7																	
8																	
9																	
10																	
11																	
12																	
13																	
14																	
15																	
16																	
17																	
18																	
19																	
20																	
21																	

The screenshot above is from the case file `andes/cases/kundur/kundur_fault.xlsx`. In this PQ workbook, there are two PQ instances (also known as "devices") called `PQ_0` and `PQ_1` (referred to by `idx`). They are connected to buses 7 and 8, respectively. The `bus` parameter of `PQ` is known as an indexing parameter (`andes.core.param.IdxParam`) through which the connections are specified. To get the connection actually work, on the `Bus` sheet, two rows must exist for two bus devices, respectively, with `idx` being 7 and 8.

To find out the `IdxParams` of a device for connecting to other devices, refer to [Model reference](#). For example, to find out how a device of the exciter model `EXDC2` should be connected to a synchronous generator, in the **Parameters** section, check the *Properties* column for *mandatory* parameters. Almost all `IdxParams` are mandatory, meaning that they must be specified to build a consistent test case. It can be seen that for `EXDC2`, `syn` is mandatory. From the description, one can tell that this is the "Synchronous generator `idx`", which should be the `idx` of an existing synchronous generator, i.e., `GENROU` or `GENCLS`.

Typically, models of the same group use the same `IdxParams` to connect to other models. Each link on the left sidebar of [Model reference](#) is a group, such as the `Exciter` group. With this convention, all exciters have a `syn` parameter for connecting to synchronous generators.

## Creating cases

It is often easier to modify from existing cases than creating from scratch. We recommend that you get familiar with the cases available with ANDES, see [Test Cases](#).

## Adding devices

Adding devices to an existing workbook is straightforward. Navigate to the sheet corresponding to the model and add a new line below the existing lines.

Almost all models have so-called mandatory parameters. They are essential to describe a complete and consistent test case. For example, the `PQ` model has the `bus` parameter as mandatory to indicate the connected bus. To look up mandatory parameters, see [Model reference](#) or use `andes doc MODEL_NAME`. Check for "mandatory" in the last column called "Properties". This column also contains other data consistency requirements discussed in the following.

Non-mandatory parameters are optional, meaning that if not provided, ANDES will use the default parameters. The default values can also be found in [Model reference](#). This does not mean that such parameters should always be left blank. For example, the `p0` (active power load) of `PQ` is optional, but likely one wants to set it to a non-zero value.

There are consistency requirements for parameters, such as `non_zero`, `non_negative` or `non_positive`. If unmet, the default values will be used. See the class reference in `andes.core.param.NumParam`.

## Autofill data

When you finished adding devices but left some optional parameters empty, you can use ANDES to autofill them. This is useful when you want to populate a large number of devices with the same parameters that can be modified later.

The autofill is done through the data converter, namely, `--convert` or `-c`. ANDES will read in the Excel file, fill the optional parameters with default values, fix the inconsistent values, and then export the data back to Excel.

**Warning:** Please backup the spreadsheet if it contains customized edits. Inconsistent data will be replaced during the conversion. Formatting in the spreadsheet will be lost. Unrecognized sheets will also be discarded.

To autofill `kundur_full.xlsx`, do

```
andes run kundur_full.xlsx -c
```

You will be prompted to confirm the overwrite.

Since this autofill feature utilizes the converter, the autofilled data can be exported to other formats, such as `.json`. To do so, use `-c json`.

## Adding workbooks

If one wants to add workbooks for models that does not exist in an `xlsx` file, one can use `--add-book ADD_BOOK` (or `-b ADD_BOOK`), where `ADD_BOOK` can be a single model name or comma-separated model names (*without space*). For example,

```
andes run kundur_full.xlsx -b Fault
```

will add an empty `Fault` sheet to `kundur_full.xlsx`.

**Warning:** With `--add-book`, the `xlsx` file will be overwritten with the same parameter corrections as in the autofill. Please make backups as needed.

Format conversion and workbook addition can be performed together. To convert a PSS/E raw file and a `dyr` file into an `xlsx` file and add a workbook for `Fault`, do

```
andes run kundur.raw -addfile kundur_full.dyr -c -b Fault
```

The output will have the same name as the raw file.

## Data Consistency

Input data needs to have consistent types for `idx`. Both string and numerical types are allowed for `idx`, but the original type and the referencing type must be the same. Suppose we have a bus and a connected PQ. The Bus device may use `1` or `'1'` as its `idx`, as long as the PQ device uses the same value for its `bus` parameter.

The ANDES `xlsx` reader will try to convert data into numerical types when possible. This is especially relevant when the input `idx` is string literal of numbers, the exported file will have them converted to numbers. The conversion does not affect the consistency of data.

## Parameter Check

The following parameter checks are applied after converting input values to array:

- Any `NaN` values will raise a `ValueError`
- Any `inf` will be replaced with  $10^8$ , and `-inf` will be replaced with  $-10^8$ .

## 1.5.2 PSS/E RAW and DYR

The Siemens PSS/E data format is a widely used for power system simulation. PSS/E uses a variety of plain-text files to store data for different actions. The RAW format (with file extension `.raw`) is used to store the steady-state data for power flow analysis, and the DYR format (with extension `.dyr`) is used to store the parameters of dynamic devices for transient simulation.

### RAW Compatibility

ANDES supports PSS/E RAW in versions 32 and 33. Newer versions of `raw` files can store PSS/E settings along with the system data, but such feature is not yet supported in ANDES. Also, manually edited `raw` files can confuse the parser in ANDES. Following manual edits, it is strongly recommended to load the data into PSS/E and save the case as a v33 RAW file.

ANDES supports most power flow models in PSS/E. It needs to be recognized that the power flow models in PSS/E is a larger set compared with those in ANDES. For example, switched shunts in PSS/E are converted to fixed ones, not all three-winding transformer flags are supported, and HVDC devices are not yet converted. This is not an exhaustive list, but all of them are advanced models.

We welcome contributions but please also reach out to us if you need to arrange the development of such models.

## DYR Compatibility

Fortunately, the DYR format does not have different versions yet. ANDES support reading parameters from DYR files for models that have been implemented in ANDES. Owing to the descriptive modeling framework, we implement the identical model so that parameters can be without conversion. If a dyr file contains models that are not recognized by ANDES, an error will be thrown. One needs to manually remove those unsupported models to load.

Like RAW files, manually edited DYR files can often be understood by PSS/E but may confuse the ANDES parser. We also recommend to load and re-save the file using PSS/E.

## Loading files

In the command line, PSS/E files can be loaded with

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

where `--addfile` or `-a` is used to specify the optional DYR file. For now, DYR files can only be added to a RAW file. We will allow different formats to be mixed in the future.

Likewise, one can convert PSS/E files to ANDES xlsx:

```
andes run kundur.raw --addfile kundur.dyr -c
```

This will convert all models in the RAW and DYR files. If only the RAW file is provided, only power flow models will be converted. One cannot easily append those in a DYR file to an existing xlx file yet.

To load PSS/E files into a scripting environment, see Example - "Working with Data".

## Creating disturbances

Instead of converting `raw` and `dyr` to `xlsx` before adding disturbances, one can edit the `.dyr` file with a plain-text editor (such as Notepad) and append lines customized for ANDES models. This is for advanced users after referring to `andes/io/psse-dyr.yaml`, at the end of which one can find the format of `Toggle`:

```
# === Custom Models ===
Toggle:
  inputs:
    - model
    - dev
    - t
```

To define two Toggles in the `.dyr` file, one can append lines to the end of the file using, for example,

```
Line  'Toggle'  Line_2  1  /
Line  'Toggle'  Line_2  1.1  /
```

which is separated by spaces and ended with a slash. The second parameter is fixed to the model name quoted by a pair of single quotation marks, and the others correspond to the fields defined in the above `inputs`. Each entry is properly terminated with a forward slash.

## Mapping dyr to ANDES models

ANDES supporting parsing PSS/E dynamic files in the format of `.dyr`. Support new dynamic models can be added by editing the input and output conversion definition file in `andes/io/psse-dyr.yaml`, which is in the standard YAML format. To add support for a new dynamic model, it is recommended to start with an existing model of similar functionality.

Consider a GENCLS entry in a dyr file. The entry looks like

```
1 'GENCLS' 1    13.0000  0.000000 /
```

where the fields are in the order of bus index, model name, generator index on the bus, inertia (H) and damping coefficient (D).

The input-output conversion definition for GENCLS is as follows

```
GENCLS:
  destination: GENCLS
  inputs:
    - BUS
    - ID
    - H
    - D
  find:
    gen:
      StaticGen:
        bus: BUS
        subidx: ID
  get:
    u:
      StaticGen:
        src: u
        idx: gen
    Sn:
      StaticGen:
        src: Sn
        idx: gen
    Vn:
      Bus:
        src: Vn
        idx: BUS
    ra:
      StaticGen:
        src: ra
        idx: gen
    xs:
      StaticGen:
```

(continues on next page)

(continued from previous page)

```

src: xs
idx: gen

outputs:
  u: u
  bus: BUS
  gen: gen
  Sn: Sn
  Vn: Vn
  D: D
  M: "GENCLS.H; lambda x: 2 * x"
  ra: ra
  xd1: xs

```

It begins with a base-level definition of the model name to be parsed from the dyr file, namely, GENCLS. Five directives can be defined for each model: destination, inputs, outputs, find and get. Note that find and get are optional, but the other three are mandatory.

- destination is ANDES model to which the original PSS/E model will be converted. In this case, the ANDES model have the same name GENCLS.
- inputs is a list of the parameter names for the PSS/E data. Arbitrary names can be used, but it is recommended to use the same notation following the PSS/E manual.
- outputs is a dictionary where the keys are the ANDES model parameter and the values are the input parameter or lambda functions that processes the inputs (see notes below).
- find is a dictionary with the keys being the temporary parameter name to store the idx of external devices and the values being the criteria to locate the devices. In the example above, GENCLS will try to find the idx of StaticGen with bus == BUS and the subidx == ID, where BUS and ID are from the dyr file.
- get is a dictionary with each key being a temporary parameter name for storing an external parameter and each value being the criteria to find the external parameter. In the example above, a temporary parameter u is the u parameter of StaticGen whose idx == gen. Note that gen is the idx of StaticGen retrieved in the above find section.

For the inputs section, one will need to skip the model name because for any model, the second field is always the model name. That is why for GENCLS below, we only list four input parameters.

```
1 'GENCLS' 1 13.0000 0.000000 /
```

For the outputs section, the order can be arbitrary, but it is recommended to follow the input order as much as possible for maintainability. In particular, the right-hand-side of the outputs can be either an input parameter name or an anonymous expression that processes the input parameters. For the example of GENCLS, since ANDES internally uses the parameter of M = 2H, the input H needs to be multiplied by 2. It is done by the following

```
M: "GENCLS.H; lambda x: 2 * x"
```

where the left-hand-side is the output parameter name (destination ANDES model parameter name), and the right-hand-side is arguments and the lambda function separated by semi-colon, all in a pair of double quotation

marks. Multiple arguments are accepted and should be separated by comma. Arguments can come from the same model or another model. In the case of the same model, the model name can be neglected, namely, by writing M: "H; lambda x: 2 \* x".

### 1.5.3 MATPOWER

ANDES supports MATPOWER data in version 2 in part for power flow calculation. The following fields are supported:

- mpc.busMVA
- mpc.bus
- mpc.gen
- mpc.branch
- mpc.area
- mpc.bus\_name

Other fields are not supported, most notably, mpc.gencost.

Power flow calculation results for MATPOWER cases are typically identical to that from MATPOWER using default settings. These settings include no reactive power limits and following all generator connectivity status. Discrepancies in the power flow solution between ANDES and MATPOWER are typically due to configuration issues or different interpretation of the data, rather than in the power flow models.

### 1.5.4 ANDES JSON

#### Overview

JSON is a portable format for storing data. It has been used in several other power system tools, including PowerModels, Pandapower, NREL-SIIP, and GridCal. It must be noted that JSON files from these tools are not interoperable because JSON only defines the data structure, not the meaning of data.

Compared with the *xlsx* file which is a zipped package, the ANDES JSON file is much faster to parse. We recommend that you use JSON in the following scenarios:

- Your test case is stable and require no manual editing, or
- You will read/write a large number of cases

To convert *kundur\_full.xlsx* to the ANDES JSON format, do

```
andes run kundur_full.xlsx -c json
```

The output file will be named *kundur\_full.json*.

## Data storage

The ANDES JSON format uses one large dictionary for all devices in the system. The keys of the dictionary are the model names, and the values are lists of dictionaries. In each dictionary, the keys are the parameter names and the values are the parameter values.

The following shows the structure of a JSON file:

```
{
  "Toggle": [
    {
      "idx": 1,
      "u": 1.0,
      "name": "Toggle_1",
      "model": "Line",
      "dev": "Line_8",
      "t": 2.0
    } //      <- Toggle_1 ends
  ], //      <- Toggle model ends
  "Bus": [
    {
      "idx": 1,
      "u": 1.0,
      "name": 1,
      "Vn": 20.0,
      "vmax": 1.1,
      "vmin": 0.9,
      ... //      <- other parameters are omitted
    },
    {
      "idx": 2,
      "u": 1.0,
      "name": 2,
      "Vn": 20.0,
      "vmax": 1.1,
      "vmin": 0.9,
      ... //      <- other parameters are omitted
    },
    ... //      <- other buses
  ], //      <- Bus model ends
  ... //      <- other models
} //      <- whole system ends
```

There are thirdparty tools for editing JSON files, but we still recommend to convert files to `xlsx` for editing. The conversion can be readily done with

```
andes run kundur_full.json -c xlsx
```

## 1.5.5 Disturbances

### Disturbance Devices

Predefined disturbances at specified time can be created by adding the corresponding devices. Three types of predefined disturbances are supported:

1. Three-phase-to-ground fault on buses. See [Fault](#) for details.
2. Connectivity status toggling. Disconnecting, connecting, or reconnecting any device, including lines, generators and motors can be implemented by [Toggle](#).
3. Alteration of values. See [Alter](#) for details.

To use these devices, the time of disturbance needs to be known ahead of the simulation. The simulation program by default checks the network connectivity status after any disturbance.

### Perturbation File

One can implement any custom disturbance using a perturbation file as discussed in [Milano2010]. The perturbation file is a Python script with a function named `pert`. The example for the perturbation file can be found in `andes/cases/ieee14/pert.py`.

```
andes.cases.ieee14.pert.pert (t, system)
```

Perturbation function called at each step.

The function needs to be named `pert` and takes two positional arguments: `t` for the simulation time, and `system` for the system object. Arbitrary logic and calculations can be applied in this function to `system`.

If the perturbation event involves switching, such as disconnecting a line, one will need to set the `system.TDS.custom_event` flag to `True` to trigger a system connectivity checking, and Jacobian rebuilding and refactorization. To implement, add the following line to the scope where the event is triggered:

```
system.TDS.custom_event = True
```

In other scopes of the code where events are not triggered, do not add the above line as it may cause significant slow-down.

The perturbation file can be supplied to the CLI using the `--pert` argument or supplied to `andes.main.run()` using the `pert` keyword.

#### Parameters

**t**

[float] Simulation time.

**system**

[`andes.system.System`] System object supplied by the simulator.

## 1.6 Test Cases

ANDES ships with test cases in the `andes/cases` folder. The cases can be found in the [online repository](#).

### 1.6.1 Summary

Below is a summary of the folders and the corresponding test cases. Some folders contain a README file with notes. When viewing the case folder on GitHub, one can conveniently read the README file below the file listing.

- `smib`: single machine infinite bus (SMIB) system [[Sauer](#)].
- `5bus`: a small PJM 5-bus test case for power flow study [[PJM5](#)].
- `GBnetwork`: a 2,000-bus system for the Great Britain network [[GB](#)]. Dynamic data is randomly generated.
- `EI`: the CURENT Eastern Interconnection network [[EI](#)].
- `ieee14` and `ieee39`: the IEEE 14-bus and 39-bus test cases [[IEEE](#)].
- `kundur`: a modified Kundur's two area system from [[RLGC](#)]. The modified system is different in the number of buses and lines from the textbook.
- `matpower`: a subset of test cases from [[MATPOWER](#)].
- `nordic44`: Nordpool 44-bus test case [[Nordic](#)]. Not all dynamic models are supported.
- `npcc`: the 140-bus Northeast Power Coordinating Council (NPCC) test case originated from Power System Toolbox [[PST](#)].
- `wecc`: the 179-bus Western Electric Coordinating Council (WECC) test case [[WECC](#)].
- `wsc`: the 9-bus WSCC (succeeded by WECC) power flow data converted from [[PSAT](#)].

---

**Note:** Different systems exhibit different dynamics, thus the appropriate systems should be used to study power system stability. For example:

- The Kundur's two-area system has under-damped modes and two coherent groups of generators. It is suitable for oscillation study and transient stability studies.
  - The WECC system is known for the inter-area oscillation.
  - The IEEE 14-bus system and the 140-bus NPCC system are frequently used for frequency control studies. So is the Eastern Interconnection system.
- 

Currently, the Kundur's 2-area system, IEEE 14-bus system, NPCC 140-bus system, and the WECC 179-bus system has been verified against DSATools TSAT.

## 1.6.2 Example data

When developing models, we manually create cases with example data to verify the models. The Kundur's system and the IEEE 14-bus system are used as the bases for adding specific models. One can find many cases in the folder `andes/cases/kundur`. The case file names typically indicate the specific model added to the file. These example cases with specific models are useful when one needs to find example parameters for the model. For example:

- `kundur_ieeeeg1` indicates the use of `IEEEG1` model in a Kundur's system
- `ieee14_solar.xlsx` contains the solar PV models (`REGCA1`, `REECA1`, and `REPCA1`)
- `ieee14_plbfu1.xlsx` is the case for `PLBFU1` (playback of voltage and frequency). The case provides an example of specifying `plbfu.xlsx`
- `ieee14_timeseries.xlsx` is an example for specifying timeseries for load data, which is provided in `pqts.xlsx`

## 1.6.3 MATPOWER cases

MATPOWER cases has been tested in ANDES for power flow calculation. All following cases are calculated with the provided initial values using the full Newton-Raphson iterative approach.

### Benchmark

See [MATPOWER cases](#) for the benchmark of MATPOWER cases.

### Synthetic systems

The 70K and the USA synthetic systems have difficulties to converge using the provided initial values. One can solve the case in MATPOWER and save it as a new case for ANDES. For example, the `SyntheticUSA` case can be converted in MATLAB with

```
mpc = runpf(case_SyntheticUSA)
savecase('USA.m', mpc)
```

And then solve it with ANDES from command line:

```
andes run USA.m
```

The output should look like

```
-> Power flow calculation
Sparse solver: KLU
Solution method: NR method
Power flow initialized.
0: |F(x)| = 140.5782767
1: |F(x)| = 29.61673314
```

(continues on next page)

(continued from previous page)

```

2: |F(x)| = 4.161452394
3: |F(x)| = 0.2337870537
4: |F(x)| = 0.001149488448
5: |F(x)| = 3.646516689e-08
Converged in 6 iterations in 1.6082 seconds.

```

## 1.6.4 How to contribute

We welcome the contribution of test cases! You can make a pull request to contribute new test cases. Please follow the structure in the `cases` folder and provide an example Jupyter notebook (see `examples/demonstration`) to showcase the results of your system.

## 1.7 Performance

We discuss methods to improve the computational performance in ANDES using Numba. Performance benchmarks are also presented.

### 1.7.1 Numba compilation

In nearly all numerical simulation software, time is mostly spent on *constructing* the numerical system and *solving* it. The construction of the DAE in ANDES involves the evaluation of functions from models that implement the residuals and Jacobians.

Numba is a just-in-time compiler in Python that can turn numerical functions into compiled machine code. In ANDES, it can speed up simulations by as much as 30%. The speedup is most effective in medium-sized systems with multiple models. Such systems involve heavy function calls but rather moderate load for linear equation solvers. It is less significant in large-scale systems where solving equations is the major time consumer.

---

**Note:** Numba is supported since ANDES 1.5.0. One needs to manually install it with `python -m pip install numba` from the Anaconda Prompt.

---

### Enabling Numba JIT

Numba needs to be enabled *manually*. In the ANDES config file: in section `[System]`, set `numba = 1`, so that it looks like

```

[System]
...
numba = 1
...

```

where the . . . are other options that are omitted here.

Just-in-time compilation will compile the code upon the *first execution* based on the input types. This is the default mode of Numba. When compilation is triggered, ANDES may appear frozen due to the compilation lag. To reuse the compiled code and save compilation time for future runs, the compiled binary code will be automatically cached. The default cache folder is in `$HOME/.andes/pydata/__pycache__` with file extensions `nbc` and `nbi`.

Numba compilation needs to be distinguished from the ANDES code generation by `andes prepare`. The ANDES code generation is to generate Python code from symbolically defined models and is relatively fast. The Numba compilation further compiles the generated Python code to machine code. Whenever the ANDES code generation produces new Python code, the cached Numba binary code will be invalidated.

## When not to compile

when developing models, we recommend disabling numba to avoid spending time on compilation.

## Ahead-of-time compilation

Just-in-time compilation can feel laggy. When ANDES is not being developed, one can compile the generated Python code ahead of time to avoid just-in-time delays. We call it "precompilation".

Precompilation is invoked by

```
andes prep -c
```

It may take a minute for the first time. Owing to caching, future compilations will be incremental and much faster.

### 1.7.2 MATPOWER cases

The hardware platform is an AMD Ryzen 9 5950X with 64GB of 3200 MHz DDR4 RAM. The operating system is WSL2 Ubunbu 20.04 on Windows 10. Software packages are:

- ANDES 1.5.3
- KVXOPT 1.2.7.1
- NumPy 1.20.3
- numba 0.54.1
- OpenBLAS 0.3.18.

Numba is enabled, and all generated code are precompiled. Network connectivity checking is turned off (`[PFlow] check_conn=0`). Time to read numba cache (~0.3s) is not included.

The table below shows the power flow time in ANDES. All the cases are original in MATPOWER 7.0, and cases not listed below will not solve with ANDES 1.5.3. The computation time may vary depending on hardware, operating system, and software.

File Name	Converged?	# of Iterations	ANDES Time [s]
case1354pegase.m	1	4	0.034
case13659pegase.m	1	5	0.276
case14.m	1	2	0.009
case145.m	1	3	0.014
case15nbr.m	1	17	0.024
case17me.m	1	3	0.010
case18.m	1	3	0.011
case1888rte.m	1	2	0.025
case18nbr.m	1	18	0.026
case1951rte.m	1	3	0.031
case2383wp.m	1	6	0.059
case24_ieee_rts.m	1	4	0.012
case2736sp.m	1	4	0.053
case2737sop.m	1	5	0.060
case2746wop.m	1	4	0.053
case2746wp.m	1	4	0.054
case2848rte.m	1	3	0.043
case2868rte.m	1	4	0.056
case2869pegase.m	1	6	0.084
case30.m	1	3	0.010
case300.m	1	5	0.019
case30Q.m	1	3	0.009
case30pwl.m	1	3	0.010
case39.m	1	1	0.008
case4_dist.m	1	3	0.010
case4gs.m	1	3	0.011
case5.m	1	3	0.011
case57.m	1	3	0.010
case60nordic.m	1	1	0.008
case6468rte.m	1	6	0.144
case6470rte.m	1	4	0.111
case6495rte.m	1	5	0.130
case6515rte.m	1	4	0.116
case6ww.m	1	3	0.010
case8387pegase.m	1	3	0.143
case89pegase.m	1	5	0.015
case9.m	1	3	0.011
case9241pegase.m	1	6	0.243
case9Q.m	1	3	0.011
case9target.m	1	4	0.010
case_ACTIVSg10k.m	1	4	0.157
case_ACTIVSg200.m	1	2	0.010
case_ACTIVSg2000.m	1	3	0.042
case_ACTIVSg25k.m	1	7	0.549

continues on next page

Table 2 – continued from previous page

File Name	Converged?	# of Iterations	ANDES Time [s]
case_ACTIVSg500.m	1	3	0.015
case_ACTIVSg70k.m	1	5	1.398
case_RTS_GMLC.m	1	3	0.013
case_SyntheticUSA.m	1	5	1.727
case_ieee30.m	1	2	0.008

## 1.8 Verification

This section presents the verification of the models and algorithms implemented in ANDES by comparing the time-domain simulation results with commercial tools.

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.

### 1.8.1 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: Trapezoidal 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 ( $E_{FD}$ ,  $E_{TERM}$ ,  $P$ ,  $Q$ ,  $\delta$ ,  $I_d$  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  $E'd$ ,  $E'q$ ,  $\psi/kd$ ,  $\psi{kq}$ ,  $\psi''_d$ , and  $\psi''_q$ ) 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.

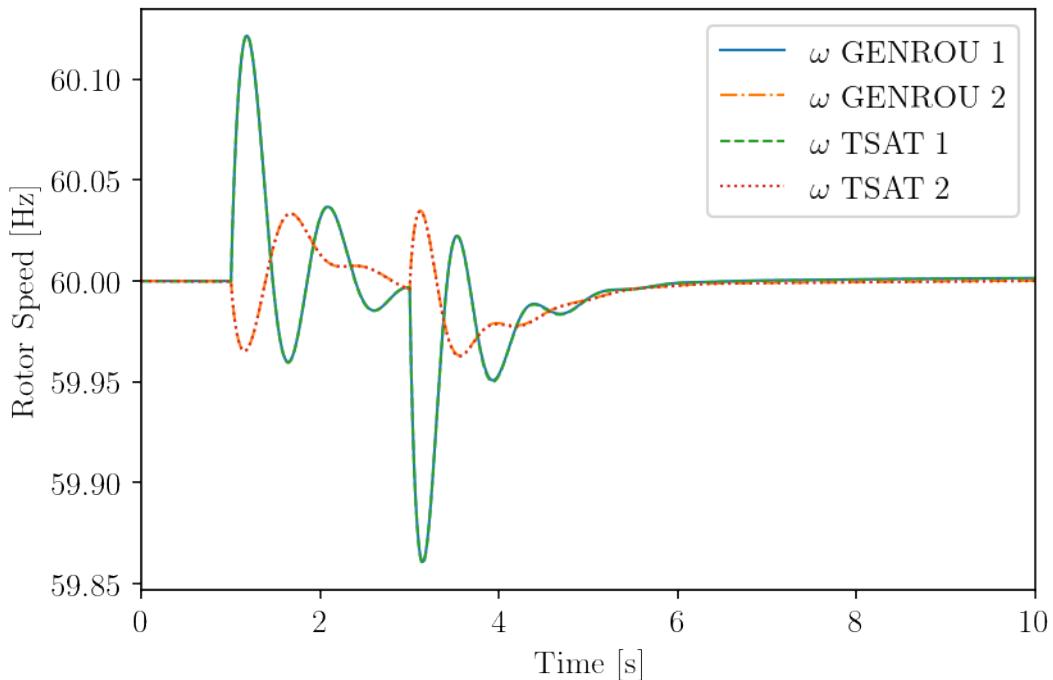
## Simulation Setup

```
ss = run_cmp('ieee14.raw', dyr='ieee14.dyr', fault_line='Line_1',
             t1=1.0, t2=3.0, tstep=1/120)
```

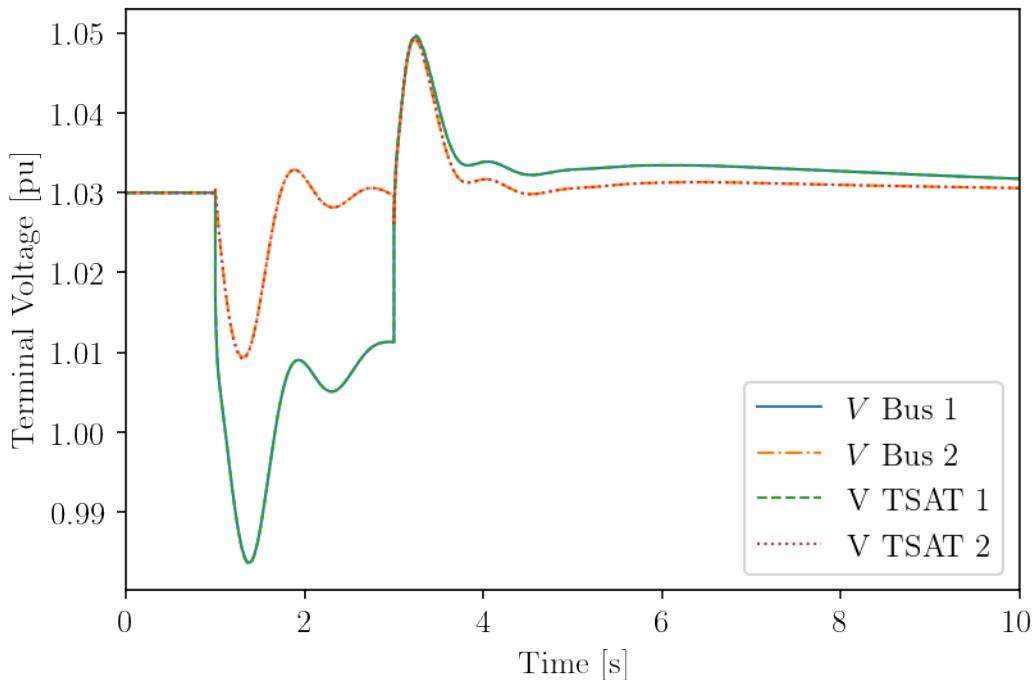
```
<Toggle Toggle_1>: Line.Line_1 status changed to 0.0 at t=1.0 sec.
<Toggle Toggle_2>: Line.Line_1 status changed to 1.0 at t=3.0 sec.
100% |██████████| 100/100 [00:16<00:00, 6.15%/s]
```

## 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]",
                           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. The line is an equivalent branch of the three-winding transformer. Tripping 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.cache.df.iloc[19]
```

idx	Line_20
u	1
name	Line_20
bus1	8
bus2	7
Sn	100
fn	60
Vn1	69

(continues on next page)

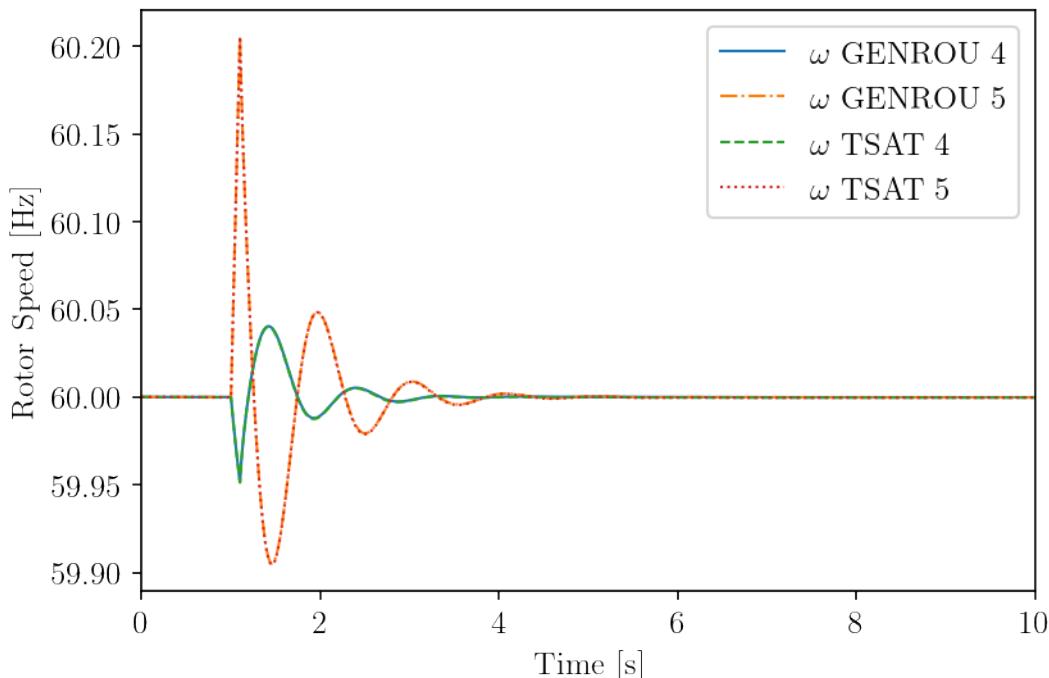
(continued from previous page)

```
Vn2           138
r              0
x            0.17615
b              0
g              0
b1             0
g1             0
b2             0
g2             0
trans          1
tap            0.99677
phi            0
owner          None
xcoord         None
ycoord         None
Name: 19, dtype: object
```

## 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)
```

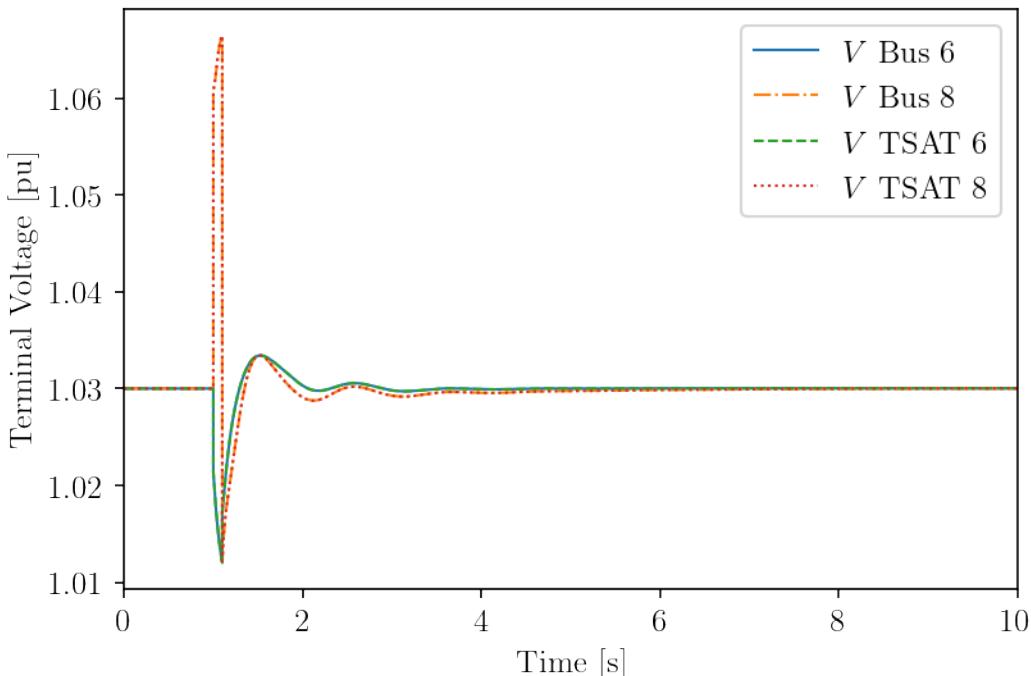


```

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

```

(continues on next page)

(continued from previous page)

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
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, 0.000, 1,1						
3,'1 ',1, 1, 1,	50.000,	25.000,	0.000,	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, 0.000, 1,1						
5,'1 ',1, 1, 1,	7.600,	1.600,	0.000,	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, 0.000, 2,1						
9,'1 ',1, 2, 2,	29.500,	16.600,	0.000,	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, 0.000, 2,1						
11,'1 ',1, 2, 2,	3.500,	1.800,	0.000,	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, 0.000, 2,1						
13,'1 ',1, 2, 2,	13.500,	5.800,	0.000,	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, 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, ↴					

(continues on next page)

(continued from previous page)

```

→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,   ↵
→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

```

(continues on next page)

(continued from previous page)

```

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
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	/			

(continues on next page)

(continued from previous page)

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	/	
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	'IEEEEST'	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

(continues on next page)

(continued from previous page)

99.000	0.0000	/
8 'TGOV1' 1	0.50000E-01	0.50000E-01 1.0500 0.30000
1.0000	2.1000	0.0000 /

## 1.8.2 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: Trapezoidal 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\$, \$\delta\$, \$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\$, \$\psi\_{kd}\$, \$\psi\_{kq}\$, \$\psi\_d\$, and \$\psi\_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.cache.df_in.iloc[0]
```

idx	Line_1
u	1
name	Line_1
bus1	1
bus2	2
Sn	100
fn	60
Vn1	345
Vn2	345
r	0.0004
x	0.0043
b	0.07
g	0
b1	0
g1	0
b2	0
g2	0

(continues on next page)

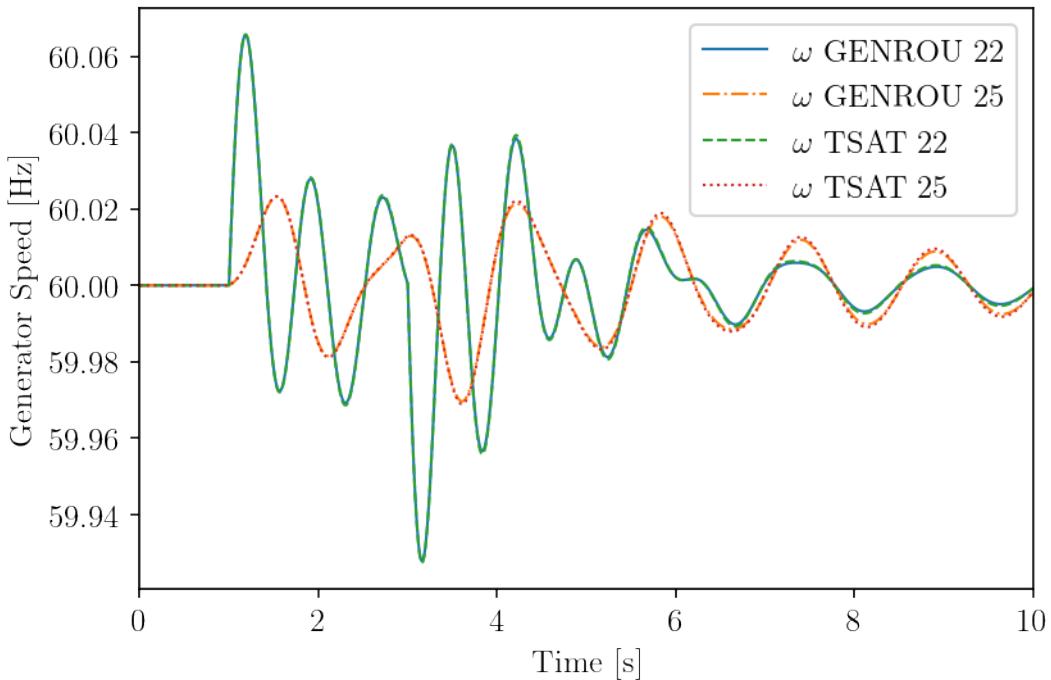
(continued from previous page)

```
trans      0
tap        1
phi        0
owner     None
xcoord    None
ycoord    None
Name: 0, dtype: object
```

## Scenario 1 Plots

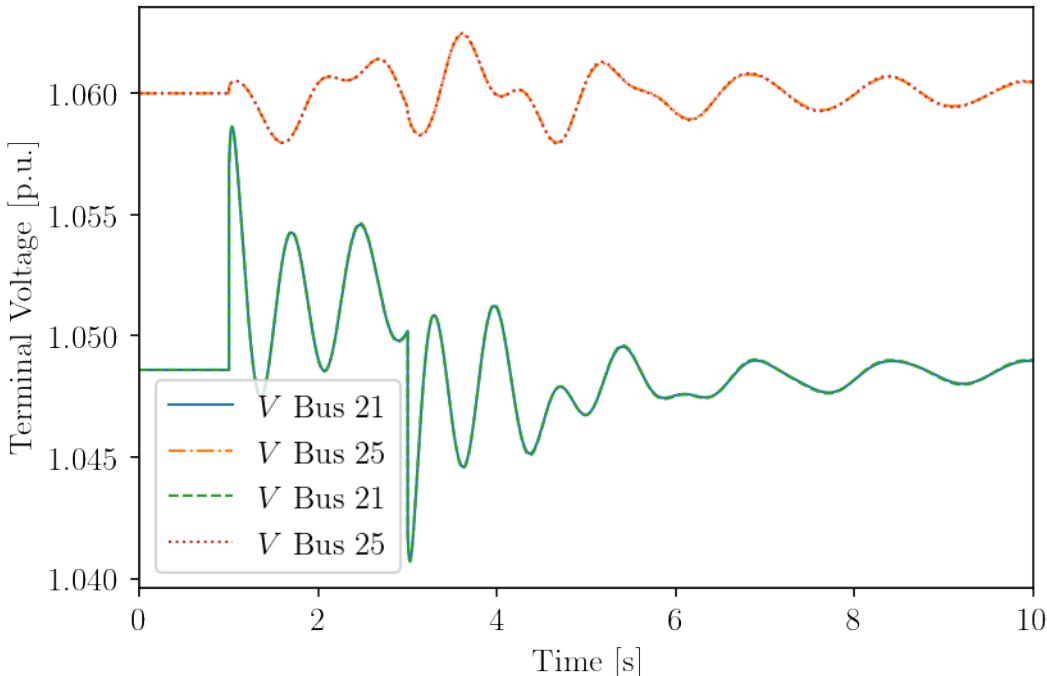
```
tsat_omega_headers = [ss.TDS.plt._fname[i + 1] for i in ss.GENROU.omega.a]
tsat_omega_headers = [i.replace('GENROU', 'TSAT') for i in tsat_omega_headers]

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
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
```

(continues on next page)

(continued from previous page)

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,'SOUTHTINGTEN	', 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
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

(continues on next page)

(continued from previous page)

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
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

(continues on next page)

(continued from previous page)

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.
↪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					

(continues on next page)

(continued from previous page)

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.
↪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					

(continues on next page)

(continued from previous page)

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.
→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					

(continues on next page)

(continued from previous page)

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.
↪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					

(continues on next page)

(continued from previous page)

```

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
  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, ↴

```

(continues on next page)

(continued from previous page)

```

→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.
→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

```

(continues on next page)

(continued from previous page)

```

  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
  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, □

```

(continues on next page)

(continued from previous page)

→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
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,

(continues on next page)

(continued from previous page)

→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
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,

(continues on next page)

(continued from previous page)

→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
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,

(continues on next page)

(continued from previous page)

→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,	0.00000,	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,	0.00000,	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,	0.00000,	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,	0.00000,	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,	0.00000,	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,	0.00000,	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,	0.00000,	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,	0.00000,	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,	0.00000,	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,	0.00000,	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,	0.00000,	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,	0.00000,	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,	0.00000,	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,	0.00000,	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,	0.00000,	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,	0.00000,	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,	0.00000,	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,	0.00000,	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,	0.00000,	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,	0.00000,	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,	0.00000,	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,	0.00000,	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,	0.00000,	0.00000,	0.00000,	0.00000,1,1,	0.00,	1,1.0000
85,	112,'1	',	6.90000E-3,	5.95000E-2,	0.10000,	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,	0.00000,	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,

(continues on next page)

(continued from previous page)

→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
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,

(continues on next page)

(continued from previous page)

→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	', 0.000000E+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	', 0.000000E+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
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,

(continues on next page)

(continued from previous page)

→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
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,

(continues on next page)

(continued from previous page)

```

→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, 0.0, 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,
→1,1.0000
 1.60000E-3, 4.35000E-2, 100.00
1.00000, 0.000, 0.000, 0.00, 0.00, 0.0, 0, 1.10000,
→0.90000, 1.10000, 0.90000, 33, 0, 0.00000, 0.00000, 0.000
1.00000, 0.000

```

(continues on next page)

(continued from previous page)

```

    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
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, -

```

(continues on next page)

(continued from previous page)

```

↪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
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

```

(continues on next page)

(continued from previous page)

```

0.00000E+0, 6.40000E-2,    100.00
1.00000,   0.000,   0.000,     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,      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,      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,      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,      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,      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,      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,      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
  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'

```

(continues on next page)

(continued from previous page)

```

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	0.23270	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	0.220955	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	0.100799	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	0.225511	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	0.23780	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	0.20930	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	0.28874	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	0.18450	0.10450	0.0000
					0.0000	/
36	'GENROU'	1	5.7000	0.30000E-01	0.35000	0.50000E-01
			4.6615	0.0000	1.9175	1.8590
			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	

(continues on next page)

(continued from previous page)

	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	
	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	/

(continues on next page)

(continued from previous page)

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
	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	/	

(continues on next page)

(continued from previous page)

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
	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

(continues on next page)

(continued from previous page)

	3.0000	1.5550	/		
26	'IEEEEX1' 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	'IEEEEX1' 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	'IEEEEX1' 1	0.0000	62.000	0.50000E-01	0.0000
	0.0000	1.0000	-1.0000	-0.50000E-01	0.40500
	0.111400	0.50000	0.0000	2.0000	0.16000E-02
	3.0000	1.7300	/		
42	'IEEEEX1' 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	'IEEEEX1' 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	'IEEEEX1' 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	'IEEEEX1' 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	'IEEEEX1' 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	'IEEEEX1' 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	'IEEEEX1' 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	'IEEEEX1' 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	'IEEEEX1' 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	1.6670	/		
57	'IEEEEX1' 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

(continues on next page)

(continued from previous page)

	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	/		

### 1.8.3 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, IEEEEST, ST2CUT and ESDC2A, the saturation of which could be implemented differently across software.

#### Simulation Parameters

Integretion method: Trapezoidal 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 ( $\$E_{FD}$ ,  $\$E_{TERM}$ ,  $\$P$ ,  $\$Q$ ,  $\$delta$ ,  $\$I_d$  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  $\$E'd$ ,  $\$E'q$ ,  $\$psi/kd$ ,  $\$psi{kq}$ ,  $\$psi''_d$ , and  $\$psi''_q$ ) 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. This branch 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.cache.df_in.iloc[220]
```

idx	Line_221
u	1
name	Line_221
bus1	38

(continues on next page)

(continued from previous page)

```

bus2          39
Sn           100
fn            60
Vn1          230
Vn2           18
r             0.0005
x             0.0238
b              0
g              0
b1             0
g1             0
b2             0
g2             0
trans          1
tap            1
phi            0
owner          None
xcoord         None
ycoord         None
Name: 220, dtype: object

```

```

# 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],

```

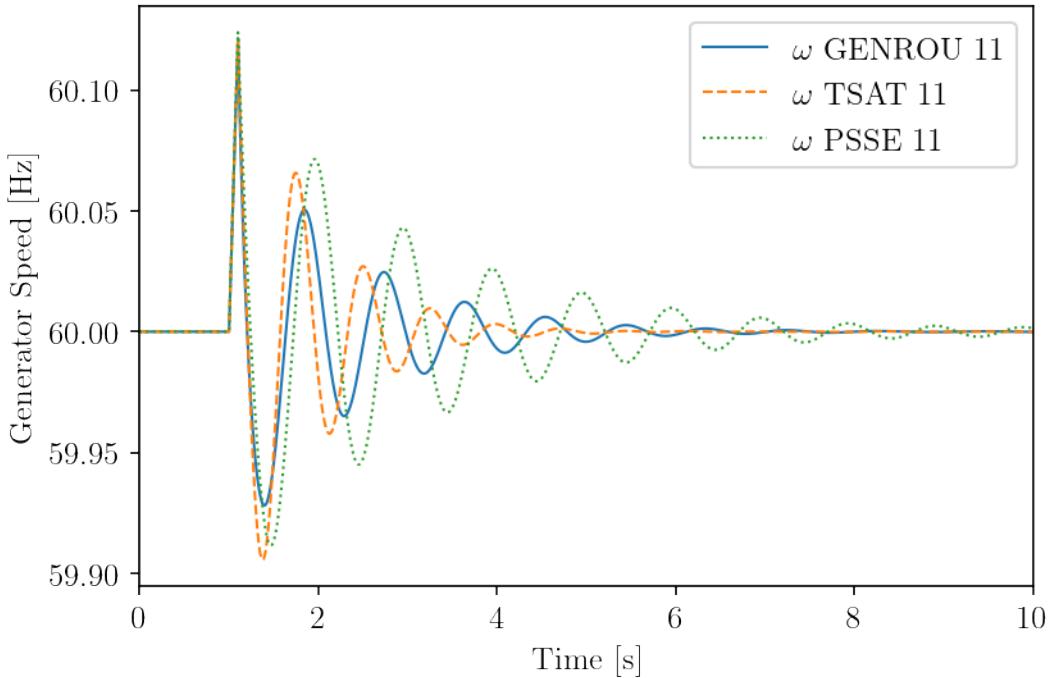
(continues on next page)

(continued from previous page)

```

ylabel="Generator Speed [Hz]",
tsat_header=tsat_omega_headers,
psse_data = omega2_psse,
psse_header = psse_omega_headers,
scale=60, right=10,
legend=True,
show=False,
)

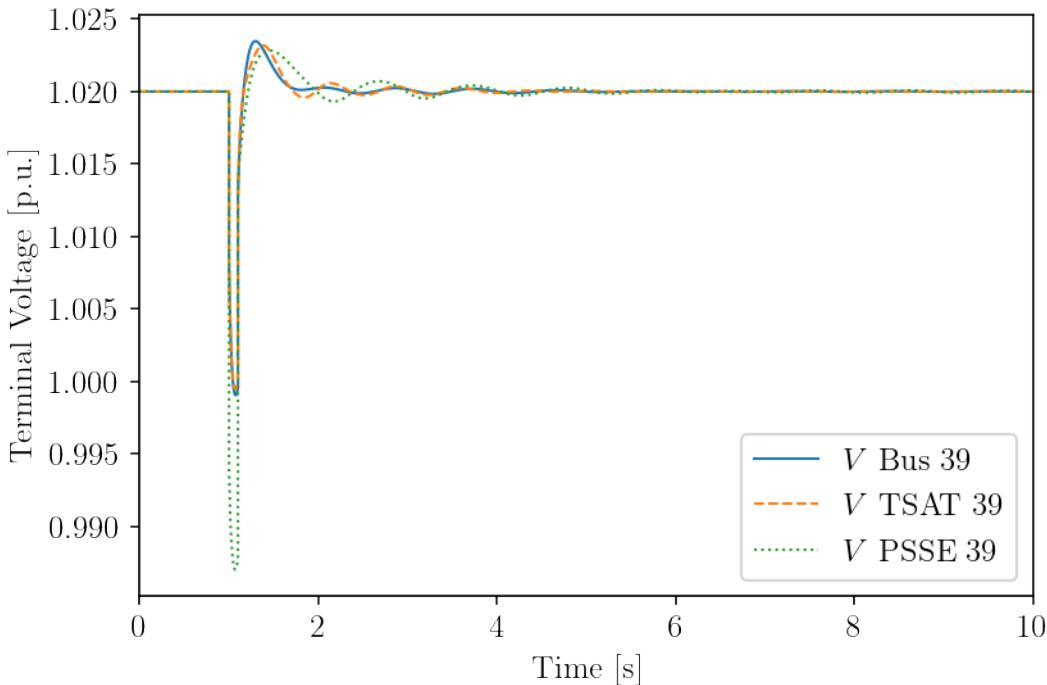
```



```

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

In this scenario, Line 4-16 trips and reconnects after 0.1 sec. This line 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.cache.df_in.iloc[1]
```

idx	Line_2
u	1
name	Line_2
bus1	4
bus2	16
Sn	100
fn	60
Vn1	345
Vn2	345
r	0.00977
x	0.11
b	2

(continues on next page)

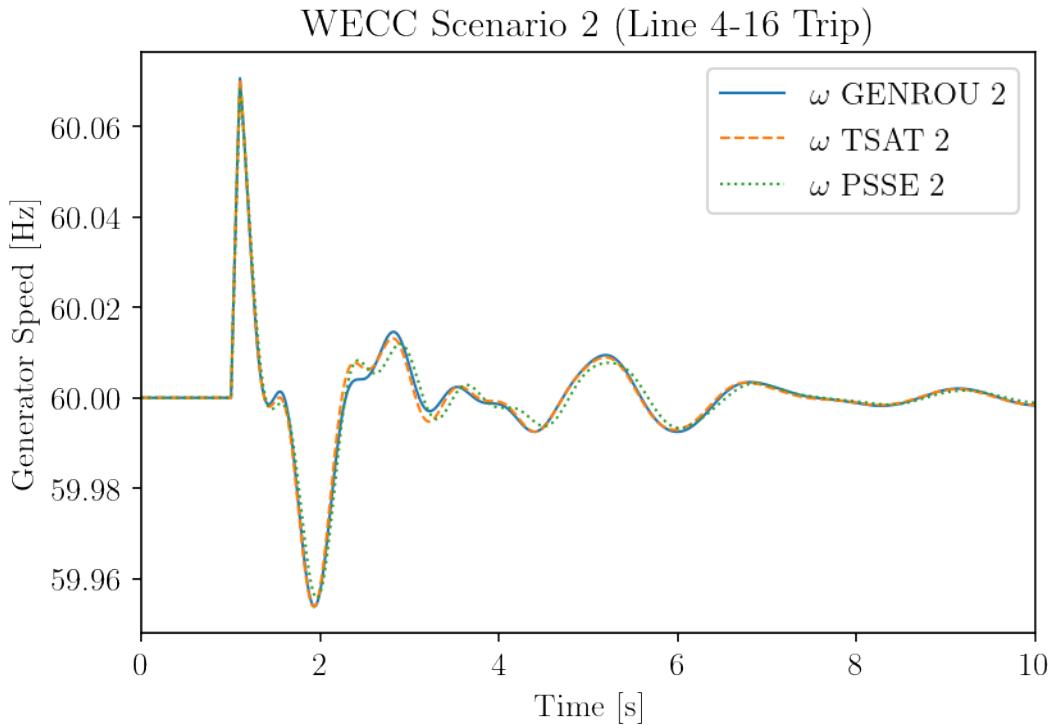
(continued from previous page)

```
g          0
b1         0
g1         0
b2         0
g2         0
trans      0
tap        1
phi        0
owner      None
xcoord    None
ycoord    None
Name: 1, dtype: object
```

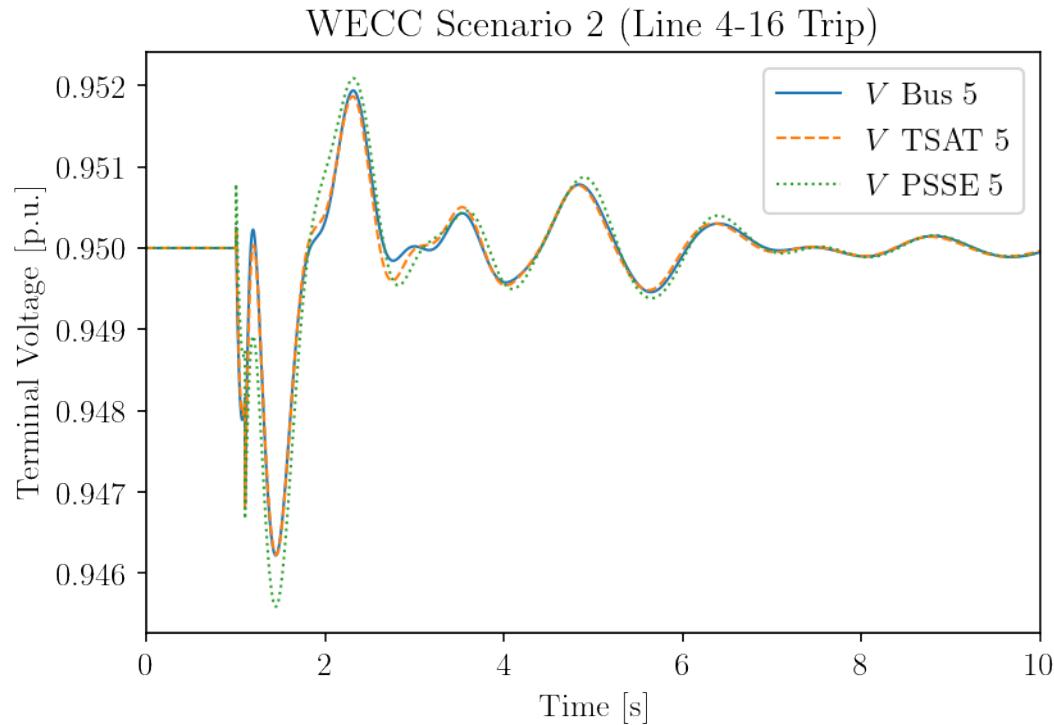
```
# load data
omega = tsat_to_df('omega.xls')
v = tsat_to_df('v.xls')

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)",
                           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: CURRENT 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, 21 MAY 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,'LOSBAÑOS	', 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, 0.000, 1,1							
3,'BL',1, 1, 1, 100.000,	0.000,	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, 0.000, 1,1							
5,'BL',1, 1, 1, 100.000,	0.000,	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, 0.000, 1,1							
8,'BL',1, 1, 1, 100.000,	0.000,	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, 0.000, 1,1							
10,'BL',1, 1, 1, 100.000,	0.000,	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, 0.000, 1,1							
12,'BL',1, 2, 1, 100.000,	0.000,	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, 0.000, 1,1							
15,'BL',1, 2, 1, 793.400,	207.000,	0.000,	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, 0.000, 1,1							
17,'BL',1, 1, 1, 100.000,	0.000,	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, 0.000, 1,1							
29,'BL',1, 1, 1, 100.000,	0.000,	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, 0.000, 1,1							
33,'BL',1, 1, 1, 3600.000,	700.000,	0.000,	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, 0.000, 1,1							
35,'BL',1, 1, 1, 100.000,	0.000,	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, 0.000, 1,1							
39,'BL',1, 2, 1, 100.000,	0.000,	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,
→0.00,	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,	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,	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,	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,	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,	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,	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,	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,	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,	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,	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,	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,
→0.00,	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,	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,	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,	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,	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,	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,	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,	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,	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,	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,	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,	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,	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,	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,	0, 1.50000,	0
↳	↳	',1,	1,1.0000			0
0.00000E+0,	1.00000E-2,	100.00				0
1.10000,	0.000,	0.000,	0.00,	0.00,	0,	0
↳0.51000,	1.50000,	0.51000,	159,	0,	0.00000,	0.000
1.00000,	0.000					0
33,	34,	0,'1 ',1,1,1,	0.00000E+0,	0.00000E+0,	0, 1.50000,	0
↳	↳	',1,	1,1.0000			0
0.00000E+0,	2.00000E-3,	100.00				0
1.00000,	0.000,	0.000,	0.00,	0.00,	0,	0
↳0.51000,	1.50000,	0.51000,	159,	0,	0.00000,	0.000
1.00000,	0.000					0
84,	35,	0,'1 ',1,1,1,	0.00000E+0,	0.00000E+0,	0, 1.50000,	0
↳	↳	',1,	1,1.0000			0
0.00000E+0,	4.60000E-3,	100.00				0
1.00000,	0.000,	0.000,	0.00,	2000.00,	0,	0
↳0.51000,	1.50000,	0.51000,	159,	0,	0.00000,	0.000
1.00000,	0.000					0
37,	47,	0,'1 ',1,1,1,	0.00000E+0,	0.00000E+0,	0, 1.50000,	0
↳	↳	',1,	1,1.0000			0
1.30000E-4,	6.93000E-3,	100.00				0
1.05000,	0.000,	0.000,	0.00,	0.00,	0,	0
↳0.51000,	1.50000,	0.51000,	159,	0,	0.00000,	0.000
1.00000,	0.000					0
38,	39,	0,'1 ',1,1,1,	0.00000E+0,	0.00000E+0,	0, 1.50000,	0
↳	↳	',1,	1,1.0000			0
5.00000E-4,	2.38000E-2,	100.00				0
1.00000,	0.000,	0.000,	0.00,	0.00,	0,	0
↳0.51000,	1.50000,	0.51000,	159,	0,	0.00000,	0.000
1.00000,	0.000					0
41,	42,	0,'1 ',1,1,1,	0.00000E+0,	0.00000E+0,	0, 1.50000,	0
↳	↳	',1,	1,1.0000			0
5.80000E-4,	2.53500E-2,	100.00				0
1.04910,	0.000,	0.000,	0.00,	0.00,	0,	0
↳0.51000,	1.50000,	0.51000,	159,	0,	0.00000,	0.000
1.00000,	0.000					0
43,	44,	0,'1 ',1,1,1,	0.00000E+0,	0.00000E+0,	0, 1.50000,	0
↳	↳	',1,	1,1.0000			0
0.00000E+0,	5.20000E-3,	100.00				0
1.02500,	0.000,	0.000,	0.00,	0.00,	0,	0
↳0.51000,	1.50000,	0.51000,	159,	0,	0.00000,	0.000
1.00000,	0.000					0
47,	46,	0,'1 ',1,1,1,	0.00000E+0,	0.00000E+0,	0, 1.50000,	0
↳	↳	',1,	1,1.0000			0
4.99000E-3,	1.14730E-1,	100.00				0
1.04780,	0.000,	0.000,	0.00,	0.00,	0,	0
↳0.51000,	1.50000,	0.51000,	159,	0,	0.00000,	0.000
1.00000,	0.000					0
48,	47,	0,'1 ',1,1,1,	0.00000E+0,	0.00000E+0,	0, 1.50000,	0
↳	↳	',1,	1,1.0000			0
2.60000E-4,	1.38600E-2,	100.00				0
1.05000,	0.000,	0.000,	0.00,	0.00,	0,	0

(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,	0, 1.0000	'	
↳	',1,	1,1.0000				'	
0.00000E+0,	9.80000E-3,	100.00					
1.05000,	0.000,	0.000,	0.00,	0.00,	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,	0, 1.0000	'	
↳	',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,	'	
↳	0.51000,	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,	0, 1.0000	'	
↳	',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,	'	
↳	0.51000,	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,	0, 1.0000	'	
↳	',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,	'	
↳	0.51000,	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,	0, 1.0000	'	
↳	',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,	'	
↳	0.51000,	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,	0, 1.0000	'	
↳	',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,	'	
↳	0.51000,	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,	0, 1.0000	'	
↳	',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,	'	
↳	0.51000,	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,	0, 1.0000	'	
↳	',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,	'	
↳	0.51000,	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,	0, 1.0000	'	
↳	',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,	'	

(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,      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,      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,      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,      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
			0.45300	0.19500	0.14500	1.9714
3	'IEEEEST'	1	3	0	0.0000	6.9000
			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
	0.0000	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
	0.0000	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
	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
	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
	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
	0.0000	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
	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
	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
	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
	0.0000	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
	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
	0.0000	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
	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
	-0.40000E-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
	0.0000	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
	0.0000	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
	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
	-0.60000E-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
	0.0000	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
	0.0000	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
	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
	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	40.000	0.20000E-01	9999.0	-9999.0
	0.0000	0.10000E-01	1.0000	/	
102	'IEEEG1'	1	0	0	20.000
	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

(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	'IEEEEST'	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	'IEEG1'	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	'IEEEEST'	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	'IEEG1'	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	'IEEEEST'	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	'IEEG1'	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
	0.20000E-01	50.000	0.20000E-01	9999.0	-9999.0
	0.0000	0.10000E-01	1.2000	/	
137	'IEEEG1'	1	0	0	20.000
	0.0000	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
	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
	0.0000	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
	0.0000	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
	0.0000	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
	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
	0.50000	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
	0.20000E-01	50.000	0.19000	9999.0	-9999.0
	0.0000	0.15000E-01	1.0000	/	
143	'IEEEG1'	1	0	0	20.000
	0.0000	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
	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
	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

(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
	0.0000	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
	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
	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
	0.20000E-01	40.000	0.20000	9999.0	-9999.0
	0.0000	0.35000E-01	1.7500	/	
148	'IEEEG1'	1	0	0	20.000
	0.0000	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
	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
	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
	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
	0.0000	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
	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
	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
	0.20000E-01	50.000	0.10000	9999.0	-9999.0
	0.0000	0.12000E-01	1.0000	/	
161	'IEEEG1'	1	0	0	20.000
	0.0000	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	/	

## 1.9 Miscellaneous

### 1.9.1 Per Unit System

The bases for AC system are

- $S_b^{ac}$ : three-phase power in MVA. By default,  $S_b^{ac} = 100MVA$  (set by `System.config.mva`).
- $V_b^{ac}$ : phase-to-phase voltage in kV.
- $I_b^{ac}$ : current base  $I_b^{ac} = \frac{S_b^{ac}}{\sqrt{3}V_b^{ac}}$

The bases for DC system are

- $S_b^{dc}$ : power in MVA. It is assumed to be the same as  $S_b^{ac}$ .
- $V_b^{dc}$ : voltage in kV.

Some device parameters are given as per unit values under the device base power and voltage (if applicable). For example, the Line model `andes.models.line.Line` has parameters `r`, `x` and `b` as per unit values in the device bases `Sn`, `Vn1`, and `Vn2`. It is up to the user to check data consistency. For example, line voltage bases are typically the same as bus nominal values. If the `r`, `x`, and `b` are meant to be per unit values under the system base, each Line device should use an `Sn` equal to the system base `mva`.

Parameters in the device base will have a property value in the Model References page. For example, `Line.r` has a property `z`, which means it is a per unit impedance in the device base. To find out all applicable properties, refer to the "Other Parameters" section of `andes.core.param.NumParam`.

After setting up the system, these parameters will be converted to per-units in the bases of system base MVA and bus nominal voltages. The parameter values in the system base will be stored to the `v` attribute of the `NumParam`. The original inputs in the device base will be moved to the `vin` attribute. For example, after setting up the system, `Line.x.v` is the line reactances in per unit under the system base.

Values in the `v` attribute is what get utilized in computation. Writing new values directly to `vin` will not affect the values in `v` afterward. To alter parameters after setting up, refer to example notebook 2.

## 1.9.2 Notes

### Modeling Blocks

#### State Freeze

State freeze is used by converter controllers during fault transients to fix a variable at the pre-fault values. The concept of state freeze applies to both state and algebraic variables. For example, in the renewable energy electric control model (REECA), the proportional-integral controllers for reactive power error and voltage error are subject to state freeze when voltage dip is observed. The internal and output states should be frozen when the freeze signal turns one and frees when the signal turns back to zero.

Freezing a state variable can be easily implemented by multiplying the freeze signal with the right-hand side (RHS) of the differential equation:

$$T\dot{x} = (1 - z_f) \times f(x)$$

where  $f(x)$  is the original RHS of the differential equation, and  $z_f$  is the freeze signal. When  $z_f$  becomes zero the differential equation will evaluate to zero, making the increment zero.

Freezing an algebraic variable is more difficult to implement. One might consider a similar solution to freezing a differential variable by constructing a piecewise equation, for example,

$$0 = (1 - z_f) \times g(y)$$

where  $g(y)$  is the original RHS of the algebraic equation. One might also need to add a small value to the diagonals of `dae.gy` associated with the algebraic variable to avoid singularity. The rationale behind this implementation is to zero out the algebraic equation mismatch and thus stop incremental correction: in the frozen state, since  $z_f$  switches to zero, the algebraic increment should be forced to zero. This method, however, would not work when a dishonest Newton method is used.

If the Jacobian matrix is not updated after  $z_f$  switches to one, in the row associated with the equation, the derivatives will remain the same. For the algebraic equation of the PI controller given by

$$0 = (K_p u + x_i) - y$$

where  $K_p$  is the proportional gain,  $u$  is the input,  $x_i$  is the integrator output, and  $y$  is the PI controller output, the derivatives w.r.t  $u$ ,  $x_i$  and  $y$  are nonzero in the pre-frozen state. These derivative corrects  $y$  following the changes of  $u$  and  $x$ . Although  $x$  has been frozen, if the Jacobian is not rebuilt, the correction will still be made due to the change of  $u$ . Since this equation is linear, only one iteration is needed to let  $y$  track the changes of  $u$ . For nonlinear algebraic variables, this approach will likely give the wrong results, since the residual is pegged at zero.

To correctly freeze an algebraic variable, the freezing signal needs to be passed to an `EventFlag`, which will set an `custom_event` flag if any input changes. `EventFlag` is a `VarService` that will be evaluated at each iteration after discrete components and before equations.

### 1.9.3 Profiling Import

To speed up the command-line program, import profiling is used to break down the program loading time.

With tool `profimp`, andes can be profiled with `profimp "import andes" --html > andes_import.htm`. The report can be viewed in any web browser.

### 1.9.4 What won't work

You might have heard that PyPy is faster than CPython due to a built-in JIT compiler. Before you spend an hour compiling the dependencies, here is the fact: PyPy won't work for speeding up ANDES.

PyPy is often much slower than CPython when using CPython extension modules (see the [PyPy\\_FAQ](#)). Unfortunately, NumPy is one of the highly optimized libraries that heavily use CPython extension modules.

## 1.10 Frequently Asked Questions

### 1.10.1 Program Startup

### 1.10.2 General

Q: What is the Hybrid Symbolic-Numeric Framework in ANDES?

A: It is a modeling and simulation framework that uses symbolic computation for descriptive modeling and code generation for fast numerical simulation. The goal of the framework is to reduce the programming efforts associated with implementing complex models and automate the research workflow of modeling, simulation, and documentation.

The framework reduces the modeling efforts from two aspects: (1) allowing modeling by typing in equations, and (2) allowing modeling using modularized control blocks and discontinuous components. One only needs to describe the model using equations and blocks without having to write the numerical code to implement the computation. The framework automatically generate symbolic expressions, computes partial derivatives, and generates vectorized numerical code.

### 1.10.3 Modeling

#### Admittance matrix

Q: Where to find the line admittance matrix?

A: ANDES does not build line admittance matrix for computing line power injections. Instead, line power injections are computed as vectors on the two line terminal. This approach generalizes line as a power injection model.

Q: Without admittance matrix, how to switch out lines?

A: Lines can be switched out and in by using Toggle. See the example in `cases/kundur/kundur_full.xlsx`. One does not need to manually trigger a Jacobian matrix rebuild because Toggle automatically triggers it using the new connectivity status.

## Reference of the existing model

Q: Is there any further reference of the existing model?

A: Most of them can be found online, such as ESIG and PowerWorld.

## 1.11 License

### 1.11.1 GNU Public License v3

Copyright 2015-2022 Hantao Cui.

ANDES is free software; you can redistribute it and/or modify it under the terms of the [GNU General Public License](#) as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.

ANDES is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the [GNU General Public License](#) for more details.

## 1.12 Quick install

Working with Conda?

ANDES is available on conda-forge and can be installed with Anaconda, Miniconda, and Mambaforge:

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

Prefer pip?

ANDES can be installed via pip from [PyPI](#).

```
pip install andes
```

New to Python?

Set up a Mambaforge environment by following [Setting Up Mambaforge](#). We recommend Mambaforge on Windows and Apple Silicon for new users.

Are you a developer?

Installing from source? Looking to develop models? Check the guide in [Develop Install](#).

---

# CHAPTER TWO

---

## EXAMPLES

A collection of examples are presented to supplement the tutorial. The examples below are identical to the Jupyter Notebook in the `examples` folder of the repository [here](#). You can run the examples in a live Jupyter Notebook online using [Binder](#).

## 2.1 Simulate and Plot

### 2.1.1 Import and Setting the Verbosity Level

We first import the `andes` library and the `get_case` function that for loading test cases shipped with ANDES.

```
import andes
from andes.utils.paths import get_case
```

We can configure the verbosity level for logging (output messages) by passing a verbosity level (10-DEBUG, 20-INFO, 30-WARNING, 40-ERROR, 50-CRITICAL) to the `stream_level` argument of `andes.main.config_logger()`. Verbose level 10 is useful for getting debug output.

The logging level can be altered (as of v1.4.3) by calling `config_logger` again with new `stream_level` and `file_level`.

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

Note that the above `andes.config_logger()` is a shorthand to `andes.main.config_logger()`. If this step is omitted, the default `INFO` level (`stream_level=20`) will be used.

## 2.1.2 Run Time-Domain Simulation

### Run power flow by default

`get_case` takes a relative path to `ANDES_ROOT/andes/cases` and returns the full path, where `ANDES_ROOT` is the root folder of ANDES.

`andes.run` is the entrypoint function for loading files and running routines. It runs power flow by default and returns a `System` object.

Note: if `default_config=True`, the default concif will be used. To use your own config, remove `default_config=True`.

See the tutorial for saving and editing ANDES config.

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

```
Working directory: "/home/hacui/repos/andes/examples"
> Loaded generated Python code in "/home/hacui/.andes/pycode".
Parsing input file "/home/hacui/repos/andes/andes/cases/kundur/kundur_full.
.xlsx"...
Input file parsed in 0.2775 seconds.
System internal structure set up in 0.0322 seconds.
-> System connectivity check results:
    No islanded bus detected.
    System is interconnected.
    Each island has a slack bus correctly defined and enabled.

-> 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)| = 3.745104027e-07
Converged in 5 iterations in 0.0084 seconds.
Report saved to "kundur_full_out.txt" in 0.0020 seconds.
```

```
-> Single process finished in 0.5018 seconds.
```

## Run time-domain simulation

Run TDS by calling `TDS.run()` on the system. Note that the call must follow the power flow immediately.

The default simulation is for 20 seconds. To change if, change `config.tf` to the desired value.

```
ss.TDS.config.tf = 10 # simulate for 10 seconds
```

```
ss.TDS.run()
```

```
-> Time Domain Simulation Summary:  
Sparse Solver: KLU  
Simulation time: 0.0-10 s.  
Fixed step size: h=33.33 ms. Shrink if not converged.  
Initialization for dynamics completed in 0.0490 seconds.  
Initialization was successful.
```

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

```
<Toggle 1>: Line.Line_8 status changed to 0 at t=2.0 sec.
```

```
Simulation completed in 0.5985 seconds.  
Outputs to "kundur_full_out.lst" and "kundur_full_out.npz".  
Outputs written in 0.0104 seconds.
```

```
True
```

To check if all operations completed successfully, check `ss.exit_code`. `exit_code == 0` means that all operations were successful.

If not zero, `exit_code` indicates the number of errors caught. One will need to check output messages for errors.

```
ss.exit_code
```

```
0
```

### 2.1.3 Export and Plot Results

If you are using ANDES interactively from Jupyter Notebook or IPython, at the end of a time-domain simulation. a plotter object `ss.TDS.plt` will automatically be created.

To check if that has been created successfully (in case the detection of an interactive environment fails), check the type of `ss.TDS.plt`.

```
ss.TDS.plt
```

```
<andes.plot.TDSData at 0x7fb229d9ff70>
```

If `ss.TDS.plt` is `None`, it can be manually loaded with `ss.TDS.load_plotter()`. Otherwise, `load_plotter()` can be safely skipped.

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

## Exporting simulation data to csv

To export simulation results to a CSV file, one can use `ss.TDS.plt.export_csv()`, which takes an optional argument of the file name.

If not provided, a default file name will be assigned.

```
ss.TDS.plt.export_csv()
```

```
CSV data saved to "/home/hacui/repos/andes/examples/kundur_full_out.csv".
```

## Index-based Plotting

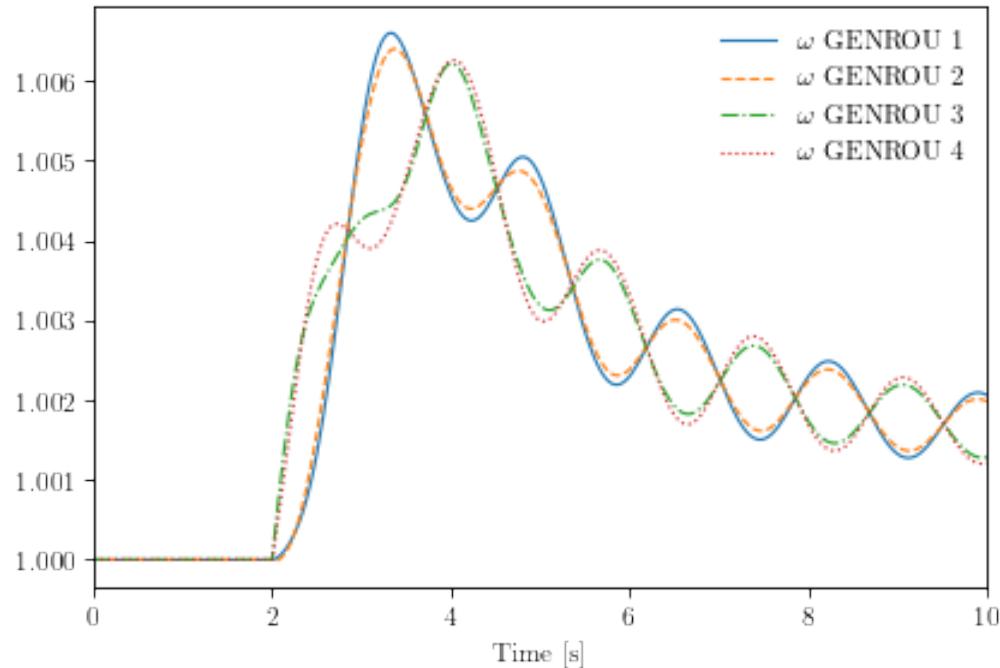
`plotter.plot()` is the entry point for plotting. It is the backend of the command-line `andes plot`.

Before plotting, open the `kundur_full_out.lst` to find the indices (first column) for the variables to plot.

For example, if we want to plot all generator speed, which is the `omega` variable of `GENROU`. By inspect, we found the indices as 5, 6, 7, 8.

Pass them in a tuple or a list to `ss.TDS.plot`.

```
fig, ax = ss.TDS.plot((5, 6, 7, 8))
```



`plot()` returns a figure object and an axis object.

### Find index by variable name

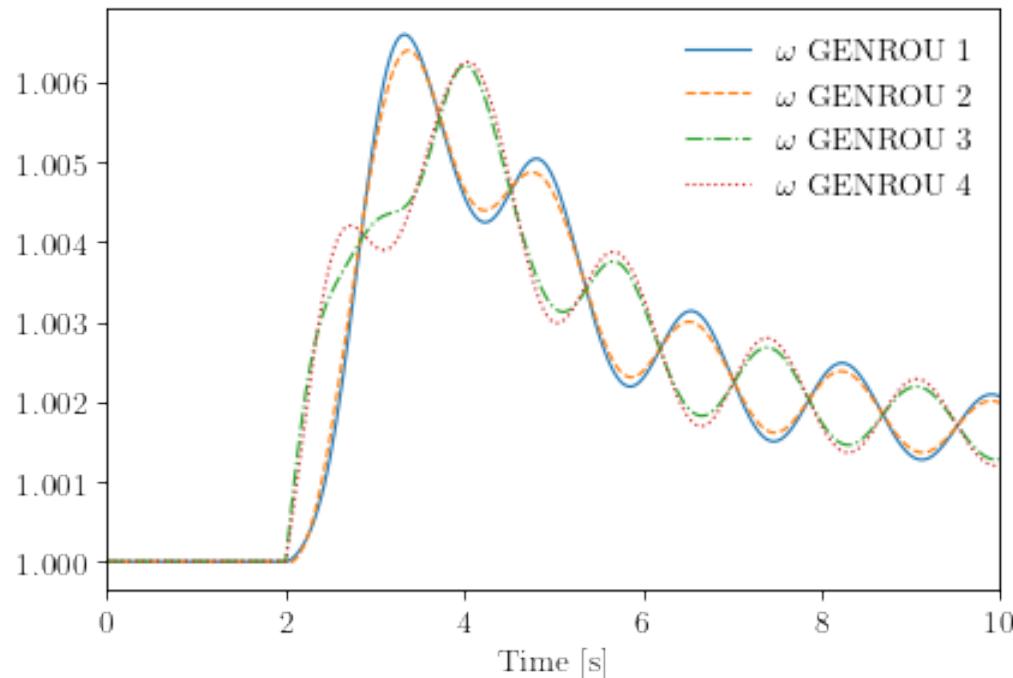
`plotter.find()` is a method for finding indices based on variable name.

The first argument is the pattern to find. An optional argument `exclude` is the pattern to exclude. Regular expression is supported for both.

```
ss.TDS.plt.find('omega')
```

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

```
fig, ax = ss.TDS.plotter.plot(ss.TDS.plotter.find('omega')[0])
```

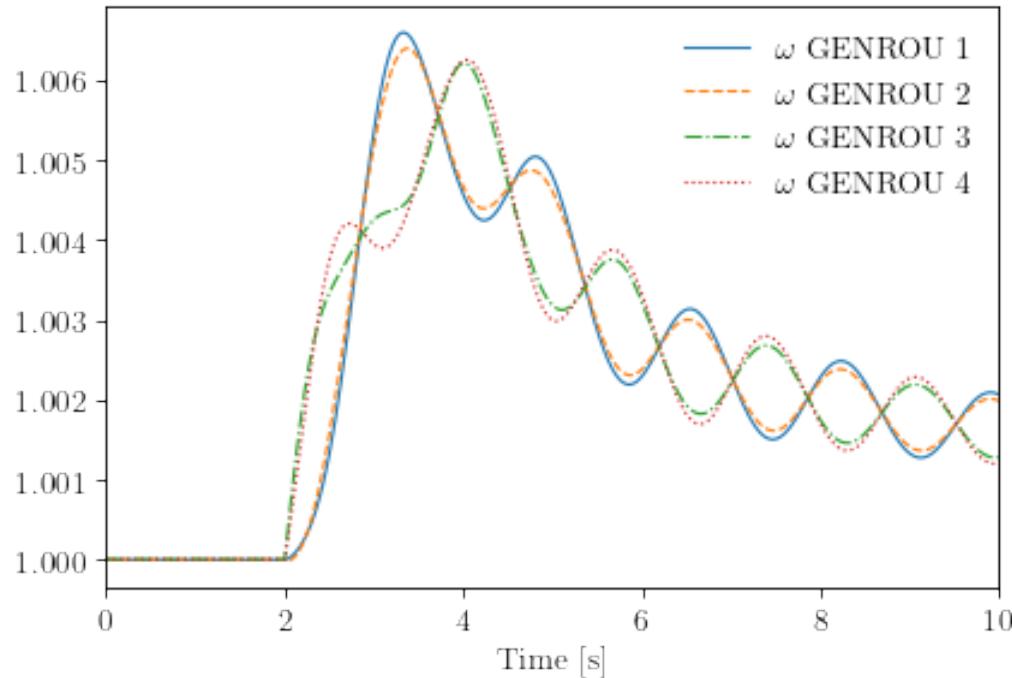


## Plotting by variable

Looking up indices from the `lst` file can be tedious.

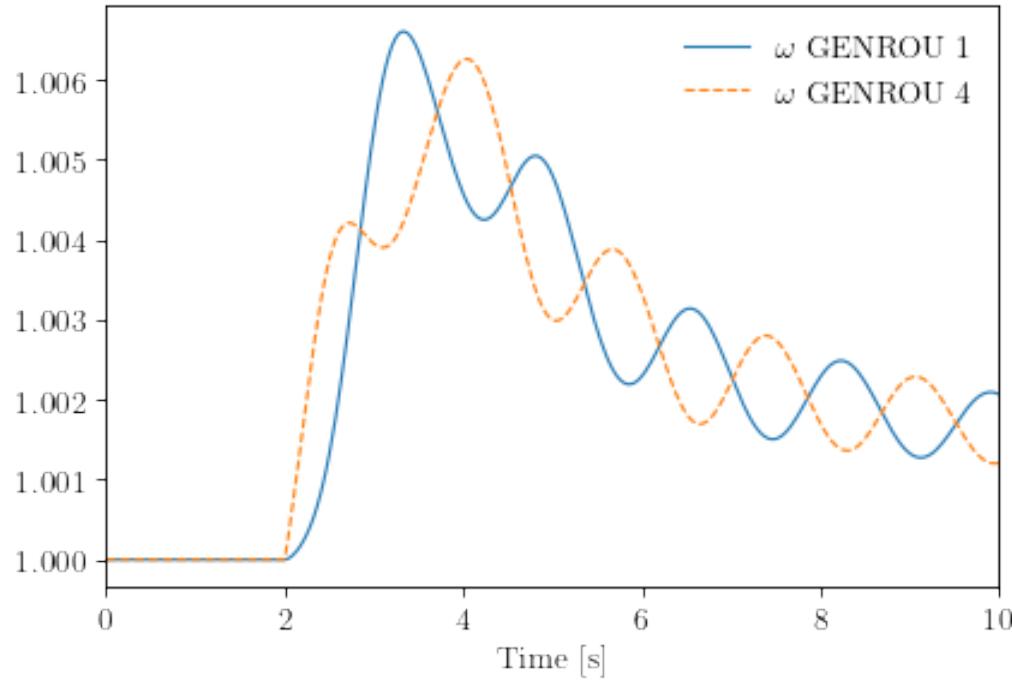
Instead, one can pass a variable in a model to `ss.TDS.plt.plot`. For example, to plot `ss.GENROU.omega`, do

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



To plot a subset of the variables, pass the 0-indexed selection indices in a tuple through argument `a` of `ss.TDS=plt.plot`. For example, to plot the 0-th and the 3-th `GENROU.omega`, do

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



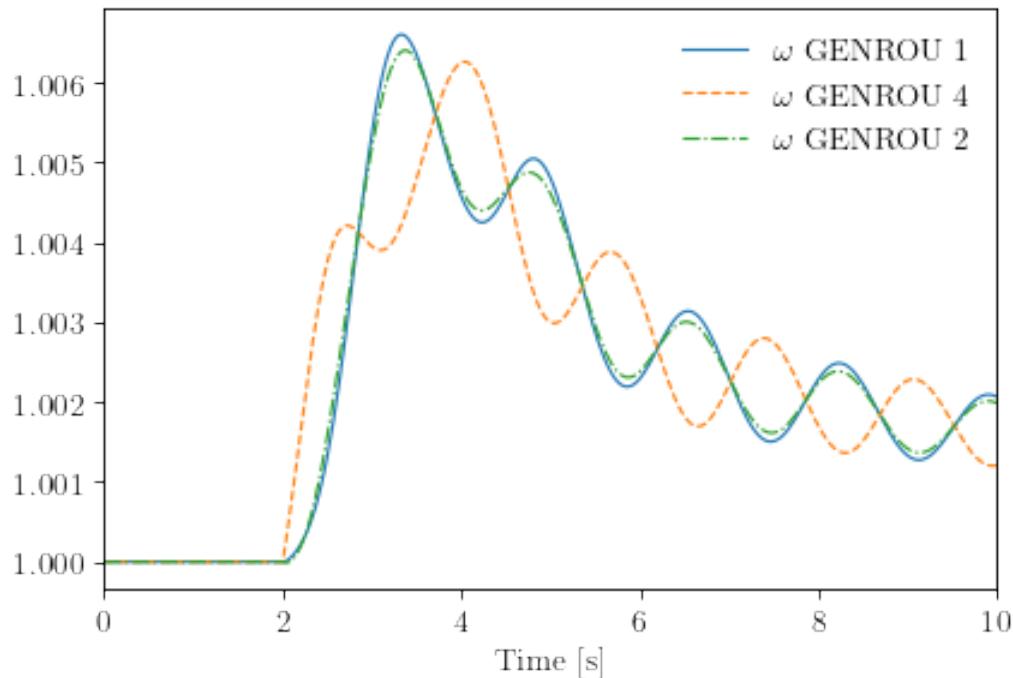
## Adding curves to an existing figure

Plotting curves into an existing figure allows easy comparison of results. It can be done by passing a figure and an axis object of `plot()`.

For example, to plot the speed of the second generator ( $a=1$ ) on the figure above, do

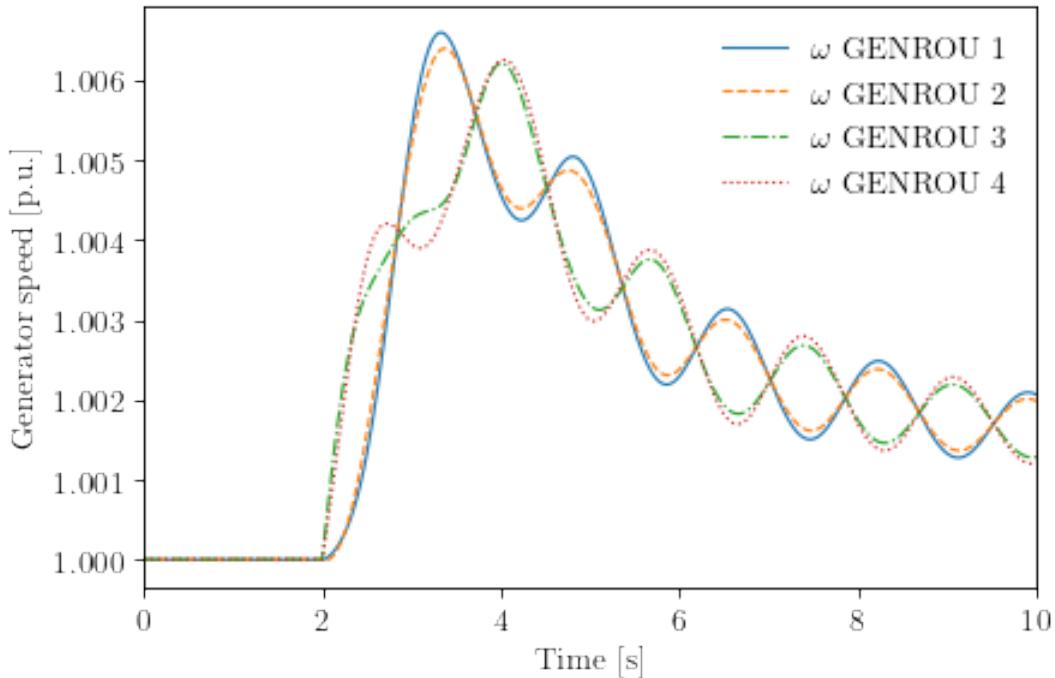
```
fig, ax = ss.TDS.plot(ss.GENROU.omega, a=(1, ), fig=fig, ax=ax,   
 ↪linestyles=['-.'])  
fig
```

<Figure size 432x288 with 0 Axes>



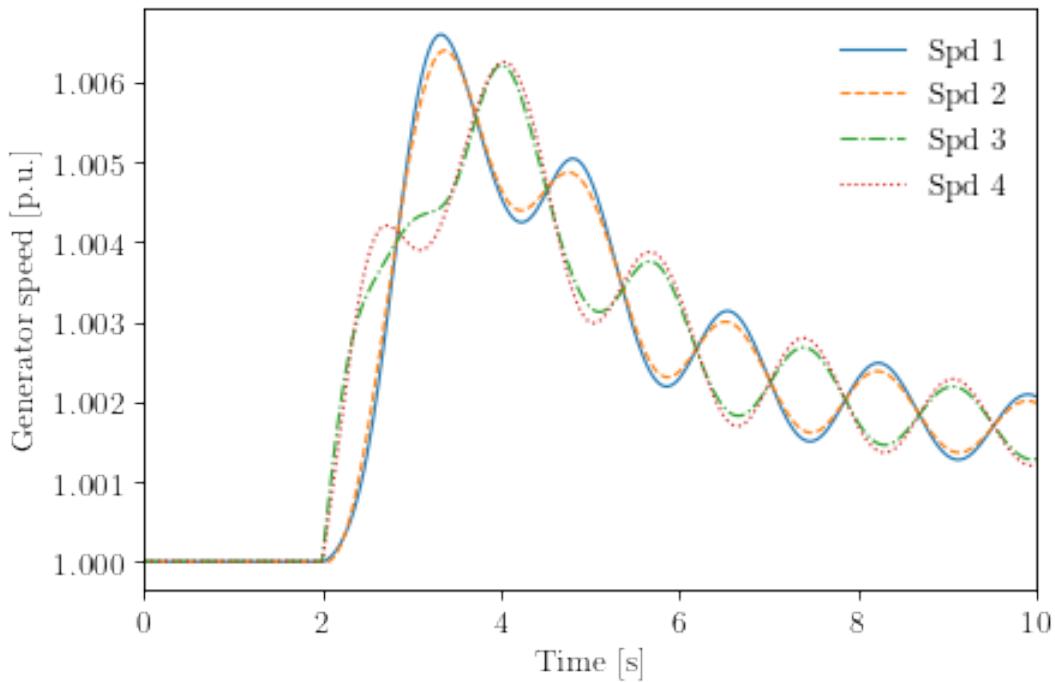
## Y-axis label

```
fig, ax = ss.TDS.plotter.plot((5, 6, 7, 8), ylabel='Generator speed [p.u.]')
```



### Legend names (yheader)

```
fig, ax = ss.TDS.plotter.plot((5, 6, 7, 8), ylabel='Generator speed [p.u.]',
                               yheader=['Spd 1', 'Spd 2', 'Spd 3', 'Spd 4'])
```



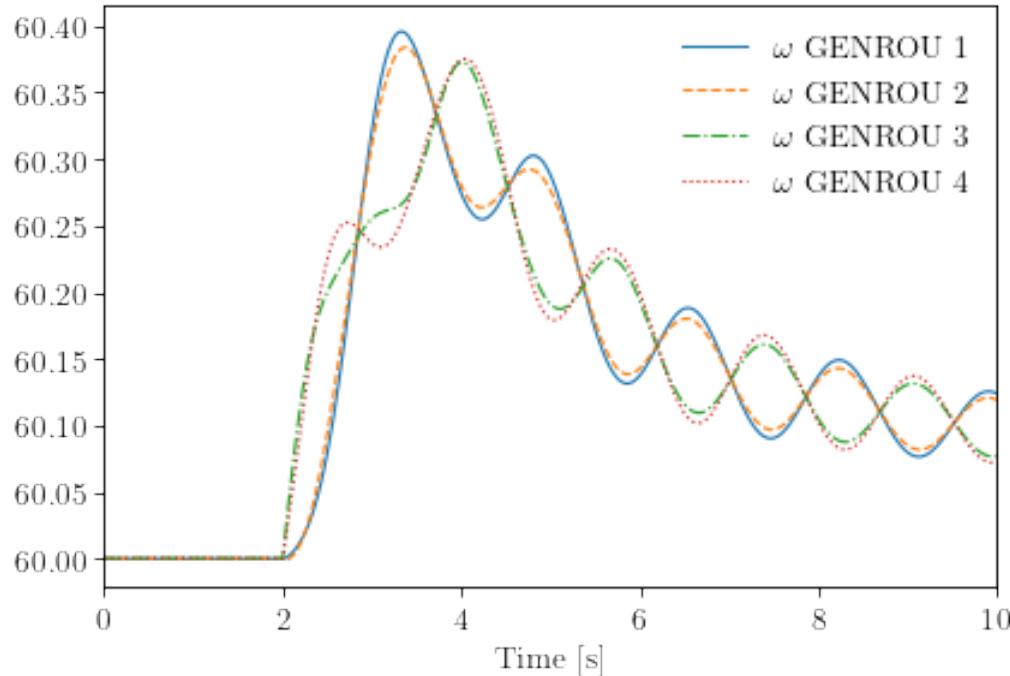
Note that the number of elements passed to `yheader` should match the number of variables. `yheader` only

applies to new curves and cannot be used to modify existing legends.

## Scaling

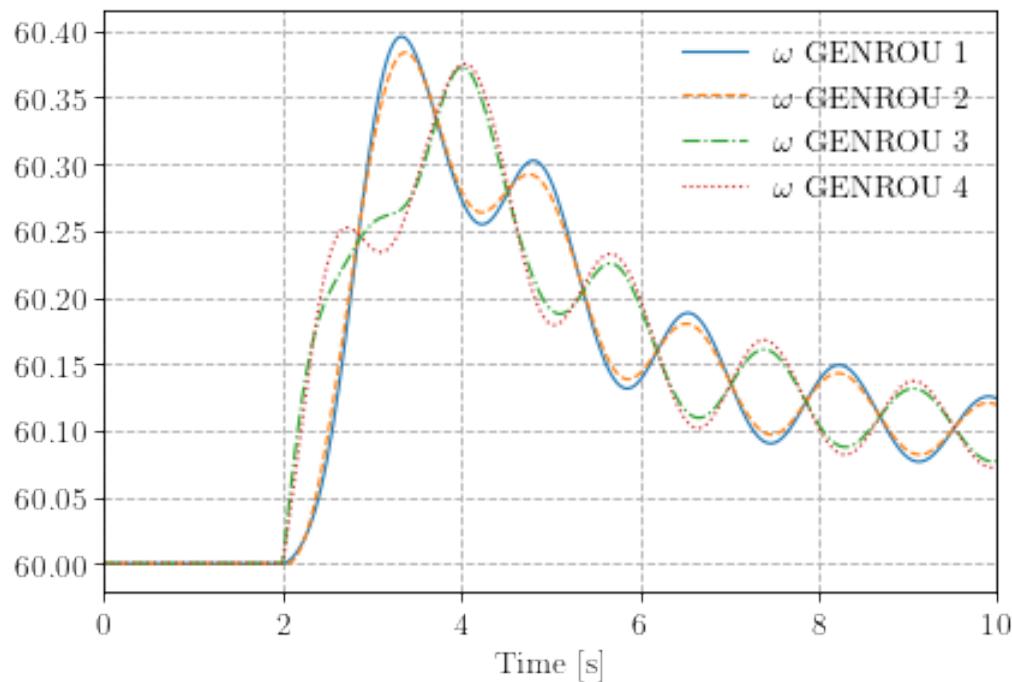
A lambda function can be passed to `yCalc` to scale the value. To scale the frequency from per unit to 60 Hz nominal values, use

```
fig, ax = ss.TDS.plotter.plot((5, 6, 7, 8), ycalc=lambda x: 60 * x)
```



## Greyscale and Grid

```
fig, ax = ss.TDS.plotter.plot((5, 6, 7, 8),
                               ycalc=lambda x: 60 * x,
                               greyscale=True,
                               grid=True)
```

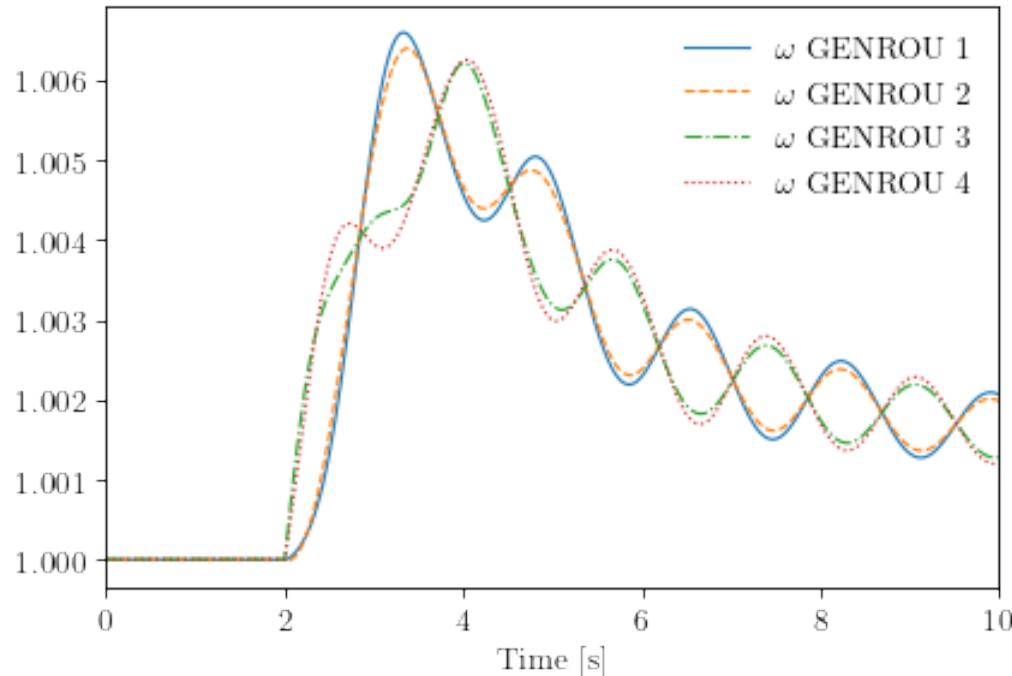


## Save figure

Pass `savefig = True` to save the figure to a png file.

```
fig, ax = ss.TDS.plotter.plot((5, 6, 7, 8), savefig=True)
```

```
Figure saved to "kundur_full_out_1.png".
```



## Additional arguments

`plotter.plot` takes addition arguments. To check additional arguments, please use `help` or refer to the source code.

```
help(ss.TDS.plotter.plot)
```

Help on method plot in module andes.plot:

```
plot(yidx, xidx=(0,), *, a=None, ytimes=None, ycalc=None, left=None, ↴
      right=None, ymin=None, ymax=None, xlabel=None, ylabel=None, xheader=None, ↴
      yheader=None, legend=None, grid=False, greyscale=False, latex=True, dpi=80, ↴
      line_width=1.0, font_size=12, savefig=None, save_format=None, show=True, ↴
      title=None, linestyles=None, use_bqplot=False, hline1=None, hline2=None, ↴
      vline1=None, vline2=None, hline=None, vline=None, fig=None, ax=None, ↴
      backend=None, set_xlim=True, set_ylim=True, autoscale=False, legend_ ↴
      bbox=None, legend_loc=None, legend_ncol=1, figsize=None, color=None, ↴
      **kwargs) method of andes.plot.TDSData instance
    Entry function for plotting.
```

This function retrieves the x and y values based on the `xidx` and `yidx` inputs, applies scaling functions `ytimes` and `ycalc` sequentially, and delegates the plotting to the backend.

Parameters

-----

`yidx` : list or int

(continues on next page)

(continued from previous page)

```

The indices for the y-axis variables
yidx : tuple or int, optional
    The index for the x-axis variable
a : tuple or list, optional
    The 0-indexed sub-indices into `yidx` to plot.
ytimes : float, optional
    A scaling factor to apply to all y values.
left : float
    The starting value of the x axis
right : float
    The ending value of the x axis
ymin : float
    The minimum value of the y axis
ymax : float
    The maximum value of the y axis
ylabel : str
    Text label for the y axis
xheader : list
    A list containing the variable names for the y-axis variable
title : str
    Title string to be shown at the top
fig
    Existing figure object to draw the axis on.
ax
    Existing axis object to draw the lines on.

```

#### Other Parameters

```

-----
ycalc: callable, optional
    A callable to apply to all y values after scaling with `ytimes`.
xlabel : str
    Text label for the x axis
xheader : list
    A list containing the variable names for the x-axis variable
legend : bool
    True to show legend and False otherwise
legend_ncol : int
    Number of columns in legend
legend_bbox : tuple of two floats
    legend box to anchor
grid : bool
    True to show grid and False otherwise
latex : bool
    True to enable latex and False to disable
greyscale : bool
    True to use greyscale, False otherwise
savefig : bool or str
    True to save to png figure file.
    str is treated as the output file name.
save_format : str
    File extension string (pdf, png or jpg) for the savefig format
dpi : int

```

(continues on next page)

(continued from previous page)

```

Dots per inch for screen print or save.
`savefig` uses a minimum of 200 dpi
line_width : float
    Plot line width
font_size : float
    Text font size (labels and legends)
figsize : tuple
    Figure size passed when creating new figure
show : bool
    True to show the image
backend : str or None
    `bqplot` to use the bqplot backend in notebook.
    None for matplotlib.
hline1: float, optional
    Dashed horizontal line 1
hline2: float, optional
    Dashed horizontal line 2
vline1: float, optional
    Dashed horizontal line 1
vline2: float, optional
    Dashed vertical line 2
hline: float or Iterable
    y-axis location of horizontal line(s)
vline: float or Iterable
    x-axis location of vertical line(s)

Returns
-----
(fig, ax)
    Figure and axis handles for matplotlib backend.
fig
    Figure object for bqplot backend.

```

## 2.1.4 Cleanup

```
! andes misc -C
```

```

"/home/hacui/repos/andes/examples/kundur_full_out.npz" removed.
"/home/hacui/repos/andes/examples/kundur_full_out.txt" removed.
"/home/hacui/repos/andes/examples/kundur_full_out.lst" removed.
"/home/hacui/repos/andes/examples/kundur_full_out.csv" removed.

```

```
!rm kundur_full_out_1.png
```

## 2.2 Working with Data

This example shows how to work with the data of a loaded test system, including parameters and variables.

```
import andes
from andes.utils.paths import get_case

andes.config_logger()
```

To show all the rows and columns, change the pandas configuration with

```
import pandas as pd

pd.options.display.max_columns = None
pd.options.display.max_rows = None
```

Let's load the Kundur's system.

### 2.2.1 Load System from an ANDES XLSX File

The ANDES xlsx file is the best supported format. Other formats can be converted to the xlsx format.

See the link below for more about format conversion. <https://github.com/cuihantao/andes/blob/master/README.md#format-converter>

As previously shown, test cases can be loaded with `andes.run()`:

```
ss = andes.run(get_case('kundur/kundur_full.xlsx'),
                default_config=True) # one can remove `default_config=True` ↵
                           ↵to use custom config file
```

```
Working directory: "/home/hacui/repos/andes/examples"
> Loaded generated Python code in "/home/hacui/.andes/pycode".
Parsing input file "/home/hacui/repos/andes/andes/cases/kundur/kundur_full.
xlsx"...
Input file parsed in 0.1394 seconds.
System internal structure set up in 0.0319 seconds.
-> System connectivity check results:
  No islanded bus detected.
  System is interconnected.
  Each island has a slack bus correctly defined and enabled.

-> 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
```

(continues on next page)

(continued from previous page)

```
4: |F(x)| = 3.745104027e-07
Converged in 5 iterations in 0.0074 seconds.
Report saved to "kundur_full_out.txt" in 0.0010 seconds.
```

```
-> Single process finished in 0.3582 seconds.
```

Alternatively, one can load a test case *without setting up* using `andes.load(..., setup=False)`. Note that `setup=False` option. It is useful to apply parameter changes to an existing test case.

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

```
Working directory: "/home/hacui/repos/andes/examples"
> Reloaded generated Python code of module "pycode".
Parsing input file "/home/hacui/repos/andes/andes/cases/kundur/kundur_full.
xlsx"...
Input file parsed in 0.1263 seconds.
```

For example, we can toggle the connectivity status `u` of `Line_3` to 0 using

```
ss.Line.alter('u', 'Line_3', 0)
```

When done, remember to set up the system before running calculation routines:

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

```
System internal structure set up in 0.0366 seconds.
-> System connectivity check results:
    No islanded bus detected.
    System is interconnected.
    Each island has a slack bus correctly defined and enabled.

-> Power flow calculation
    Numba: Off
    Sparse solver: KLU
    Solution method: NR method
Power flow initialized in 0.0049 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 0.0122 seconds.
Report saved to "kundur_full_out.txt" in 0.0016 seconds.
```

```
True
```

After setting up the system, adding or removing devices are not yet allowed.

## 2.2.2 Load System from PSS/E RAW and DYR Files

ANDES supports loading systems from PSS/E RAW and DYR files.

The PSS/E v32 raw format is best supported.

Note that this feature is experimental. We try our best to support this format, but the compatibility is not guaranteed.

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

The raw file is passed to the positional argument, whereas the dyr file is passed to addfile.

```
ss = andes.run(raw_path, addfile=dyr_path, default_config=True)
```

```
Working directory: "/home/hacui/repos/andes/examples"
> Reloaded generated Python code of module "pycode".
Parsing input file "/home/hacui/repos/andes/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 0.0047 seconds.
Parsing additional file "/home/hacui/repos/andes/andes/cases/kundur/kundur_
↪full.dyr"...
Addfile parsed in 0.0912 seconds.
System internal structure set up in 0.0310 seconds.
-> System connectivity check results:
    No islanded bus detected.
    System is interconnected.
    Each island has a slack bus correctly defined and enabled.

-> Power flow calculation
    Numba: Off
    Sparse solver: KLU
    Solution method: NR method
Power flow initialized in 0.0037 seconds.
0: |F(x)| = 3.175850023
1: |F(x)| = 3.176155228e-08
Converged in 2 iterations in 0.0044 seconds.
Report saved to "kundur_out.txt" in 0.0006 seconds.
```

```
-> Single process finished in 0.2371 seconds.
```

## Attributes for storing values

Parameters are stored as attributes of the model. For example, `ss.GENROU.M`, the machine starting time constant (2H), is stored in `ss.GENROU.M`.

```
ss.GENROU.M
```

```
NumParam: GENROU.M, v=[117.    117.    111.15 111.15], vin=[13.    13.    12.35
→12.35]
```

It is an instance of `NumParam`, which contains fields `v` for the values after converting to system-base per unit values.

```
ss.GENROU.M.v
```

```
array([117. , 117. , 111.15, 111.15])
```

And field `vin` is for the original input data.

```
ss.GENROU.M.vin
```

```
array([13. , 13. , 12.35, 12.35])
```

## Tabulated view

ANDES provides tabulated **view** of model parameters by using `DataFrame`. Each model object has an attribute called `cache` for caching the parameter dataframes.

The original parameters from the input file are stored in `cache.df_in` of the model object. For `GENROU`, do

```
ss.GENROU.cache.df_in
```

uid	idx	u	name	bus	gen	coi	coi2	Sn	Vn	fn	D	\	
0	GENROU_1	1.0	GENROU_1	1	1	None	None	900.0	20.0	60.0	0.0		
1	GENROU_2	1.0	GENROU_2	2	2	None	None	900.0	20.0	60.0	0.0		
2	GENROU_3	1.0	GENROU_3	3	3	None	None	900.0	20.0	60.0	0.0		
3	GENROU_4	1.0	GENROU_4	4	4	None	None	900.0	20.0	60.0	0.0		
uid	M	ra	xl	xd1	kp	kw	S10	S12	gammap	gammaq	xd	xq	\
0	13.00	0.0	0.06	0.3	0.0	0.0	0.0	0.0	1.0	1.0	1.8	1.7	
1	13.00	0.0	0.06	0.3	0.0	0.0	0.0	0.0	1.0	1.0	1.8	1.7	
2	12.35	0.0	0.06	0.3	0.0	0.0	0.0	0.0	1.0	1.0	1.8	1.7	
3	12.35	0.0	0.06	0.3	0.0	0.0	0.0	0.0	1.0	1.0	1.8	1.7	
uid	xd2	xq1	xq2	Td10	Td20	Tq10	Tq20						

(continues on next page)

(continued from previous page)

0	0.25	0.55	0.25	8.0	0.03	0.4	0.05
1	0.25	0.55	0.25	8.0	0.03	0.4	0.05
2	0.25	0.55	0.25	8.0	0.03	0.4	0.05
3	0.25	0.55	0.25	8.0	0.03	0.4	0.05

Parameters will be **converted** to per-unit in the system base after loading. This process have been done if `andes.run` is used for loading the data file.

To inspect the converted parameters, check the `cache.df` parameter.

```
ss.GENROU.cache.df
```

uid	idx	u	name	bus	gen	coi	coi2	Sn	Vn	fn	D	\
0	GENROU_1	1.0	GENROU_1	1	1	None	None	900.0	20.0	60.0	0.0	
1	GENROU_2	1.0	GENROU_2	2	2	None	None	900.0	20.0	60.0	0.0	
2	GENROU_3	1.0	GENROU_3	3	3	None	None	900.0	20.0	60.0	0.0	
3	GENROU_4	1.0	GENROU_4	4	4	None	None	900.0	20.0	60.0	0.0	

uid	M	ra	xl	xd1	kp	kw	S10	S12	gammap	gammaq	xd_\	\
0	117.00	0.0	0.006667	0.033333	0.0	0.0	0.0	0.0	1.0	1.0	0.2	
1	117.00	0.0	0.006667	0.033333	0.0	0.0	0.0	0.0	1.0	1.0	0.2	
2	111.15	0.0	0.006667	0.033333	0.0	0.0	0.0	0.0	1.0	1.0	0.2	
3	111.15	0.0	0.006667	0.033333	0.0	0.0	0.0	0.0	1.0	1.0	0.2	

uid	xq	xd2	xq1	xq2	Td10	Td20	Tq10	Tq20
0	0.188889	0.027778	0.061111	0.027778	8.0	0.03	0.4	0.05
1	0.188889	0.027778	0.061111	0.027778	8.0	0.03	0.4	0.05
2	0.188889	0.027778	0.061111	0.027778	8.0	0.03	0.4	0.05
3	0.188889	0.027778	0.061111	0.027778	8.0	0.03	0.4	0.05

One will notice the converted parameters such as `M`, `xl`, and all other impedances.

**It is very important to notice that `cache.df` and `cache.df_in` are both views. Altering data in these views will NOT alter the underlying parameter values.**

To alter values, see the example below.

One may have noticed that `ss.GENROU.cache.df` and `ss.GENROU.as_df()` returns the same dataframe. The difference is that the latter creates a new dataframe everytime it is called, but the former caches the dataframe when it is initially accessed.

## Altering parameters

Parameters can be altered by calling the `alter` method on a model instance.

We first look up the original value through `get`.

Either `v` or `vin` can be passed to argument `attr` to retrieve the converted or the original data. Here we are retrieving the original input data. If `attr` is not provided, `get` returns the value after per-unit conversion, which is the value used for calculation, by default.

```
ss.GENROU.get("M", "GENROU_1", attr='vin')
```

```
13.0
```

To change the `M` of `GENROU_1` to 10, do

```
ss.GENROU.alter("M", "GENROU_1", 10)
```

The value set through `alter` is always the data before per-unit conversion - just like it should have been in an input file. ANDES will perform the conversion and set `vin` and `v` correctly.

Parameters altered through `Model.alter()` can be saved as a new system using

```
andes.io.xlsx.write(ss, 'new_system.xlsx', overwrite=True)
```

```
xlsx file written to "new_system.xlsx"
```

```
True
```

## In-place update

`alter()` can be used to change the value of `ConstService` to modify the equation that depend on such `ConstService`. For example, the distributed PV model `PVD1` implements a `ConstService` called `pref0` to store the initial value of the power reference. An equation associated with variable `Pref` enforces that `0 = Pref - pref0`.

If one needs to modify `Pref`, it has to be done through modifying `pref0`. Modifying `Pref` directly will not take any effect since the variable will be overwritten by the solution of equations.

To update `pref0` for a `PVD1` device with `idx = "PVD_1"`, one can do

```
ss.PVD1.alter('pref0', 'PVD_1', 0.005)
```

or, using keyword arguments in any order,

```
ss.PVD1.alter(src='pref0', idx='PVD_1', value=0.005)
```

If `PVD_1` is the first (i.e., 0-th in the Python indexing) in the `idx` list, this modification is equivalent to setting

```
ss.PVD1.pref0.v[0] = 0.005.
```

Since index 0 is given, the array `ss.PVD1.pref0.v` is updated in-place.

When one needs to modify the `pref0` of all PVD1 devices to 0.005, one can do

```
ss.PVD1.alter('pref0', ss.PVD1.idx.v, 0.005)
```

This is equivalent to

```
ss.PVD1.pref0.v[:] = 0.005
```

Note the `[ : ]` in the above line. This is a slice operation so that the assignment happens in-place.

One must never do out-of-place assignment, i.e.,

```
ss.PVD1.pref0.v = 0.005
```

or

```
ss.PVD1.pref0.v = 0.005 * np.ones_line(ss.PVD1.pref0.v)
```

because the assignment will point `ss.PVD1.pref0.v` to a new array. Internally, ANDES reuses the memory for all arrays, meaning that their addresses are assumed to be constant. If one modifies `ss.PVD1.pref0.v` out of place, the previous memory will no longer be accessible through `ss.PVD1.pref0.v`.

On the safe side, one should modify variables using `alter()` or, at least, always use in-place assignment to internal arrays.

## Refreshing the view

As mentioned, `cache.df` and `cache.df_in` are *cached* views and will not be automatically updated for inspection.

This is generally not an issue if one performs the simulation after altering data. However, if one needs to inspect the data again, `cache.refresh()` needs to be called manually.

```
ss.GENROU.cache.refresh()
```

```
ss.GENROU.cache.df_in
```

uid		idx	u	name	bus	gen	coi	coi2	Sn	Vn	fn	D	\	
0	GENROU_1	1.0	GENROU_1		1	1	None	None	900.0	20.0	60.0	0.0		
1	GENROU_2	1.0	GENROU_2		2	2	None	None	900.0	20.0	60.0	0.0		
2	GENROU_3	1.0	GENROU_3		3	3	None	None	900.0	20.0	60.0	0.0		
3	GENROU_4	1.0	GENROU_4		4	4	None	None	900.0	20.0	60.0	0.0		
uid		M	ra	xl	xd1	kp	kw	S10	S12	gammap	gammaq	xd	xq	\
0	10.00	0.0	0.06	0.3	0.0	0.0	0.0	0.0	0.0	1.0	1.0	1.8	1.7	
1	13.00	0.0	0.06	0.3	0.0	0.0	0.0	0.0	0.0	1.0	1.0	1.8	1.7	
2	12.35	0.0	0.06	0.3	0.0	0.0	0.0	0.0	0.0	1.0	1.0	1.8	1.7	
3	12.35	0.0	0.06	0.3	0.0	0.0	0.0	0.0	0.0	1.0	1.0	1.8	1.7	
uid		xd2	xq1	xq2	Td10	Td20	Tq10	Tq20						

(continues on next page)

(continued from previous page)

0	0.25	0.55	0.25	8.0	0.03	0.4	0.05
1	0.25	0.55	0.25	8.0	0.03	0.4	0.05
2	0.25	0.55	0.25	8.0	0.03	0.4	0.05
3	0.25	0.55	0.25	8.0	0.03	0.4	0.05

Alternatively, one can call the `as_df()` function to build a new dataframe *without overwriting* the cache:

```
ss.GENROU.as_df()
```

uid	idx	u	name	bus	gen	coi	coi2	Sn	Vn	fn	D	\
0	GENROU_1	1.0	GENROU_1	1	1	None	None	900.0	20.0	60.0	0.0	
1	GENROU_2	1.0	GENROU_2	2	2	None	None	900.0	20.0	60.0	0.0	
2	GENROU_3	1.0	GENROU_3	3	3	None	None	900.0	20.0	60.0	0.0	
3	GENROU_4	1.0	GENROU_4	4	4	None	None	900.0	20.0	60.0	0.0	
uid	M	ra	xl	xd1	kp	kw	S10	S12	gammap	gammaq	xd_\	\
0	90.00	0.0	0.006667	0.033333	0.0	0.0	0.0	0.0	1.0	1.0	0.2	
1	117.00	0.0	0.006667	0.033333	0.0	0.0	0.0	0.0	1.0	1.0	0.2	
2	111.15	0.0	0.006667	0.033333	0.0	0.0	0.0	0.0	1.0	1.0	0.2	
3	111.15	0.0	0.006667	0.033333	0.0	0.0	0.0	0.0	1.0	1.0	0.2	
uid	xq	xd2	xq1	xq2	Td10	Td20	Tq10	Tq20				
0	0.188889	0.027778	0.061111	0.027778	8.0	0.03	0.4	0.05				
1	0.188889	0.027778	0.061111	0.027778	8.0	0.03	0.4	0.05				
2	0.188889	0.027778	0.061111	0.027778	8.0	0.03	0.4	0.05				
3	0.188889	0.027778	0.061111	0.027778	8.0	0.03	0.4	0.05				

## 2.2.3 Variables

### Snapshots

One might also want to check the variable values in a similar way to that for a parameter. Certainly, a variable has a `v` attribute which stores values.

**It is important to note that `v` only holds the values at the last program state.** Such program state could be the solution of power flow, the initialization of time-domain simulation, or the end of a simulation disturbances.

Since we have only ran power flow for `ss`, `ss.Bus.v.v` are the voltage magnitude solutions, where the first `v` is for "voltage", and the second `v` is the first `v`'s value attribute.

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

```
array([1.          , 1.          , 1.          , 1.          , 0.98337472,
       0.96908585, 0.9562181 , 0.95400018, 0.96856366, 0.98377143])
```

Variables hold more than values. They have an attribute `a` for the addresses indexing into the corresponding type of array.

There are two system-level arrays, `ss.dae.x` and `ss.dae.y` for the right-hand-side of the differential and algebraic equations, respectively.

```
type(ss.Bus.v)
```

```
andes.core.var.Algeb
```

`ss.Bus.v` is an algebraic variable, thus `ss.Bus.v.a` holds the indices into `ss.dae.g`.

```
ss.dae.y[ss.Bus.v.a]
```

```
array([1.          , 1.          , 1.          , 1.          , 0.98337472,
       0.96908585, 0.9562181 , 0.95400018, 0.96856366, 0.98377143])
```

We can see that these two values are the same.

## Time series

After a time-domain simulation, the time series of the variables can be retrieved through `ss.dae.ts`. Let's first run a simulation.

```
ss.TDS.run()
```

```
-> Time Domain Simulation Summary:
Sparse Solver: KLU
Simulation time: 0.0-20.0 s.
Fixed step size: h=33.33 ms. Shrink if not converged.
Initialization for dynamics completed in 0.0489 seconds.
Initialization was successful.
```

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

```
<Toggle Toggle_1>: Line.Line_8 status changed to 0 at t=2.0 sec.
```

```
Simulation completed in 1.1192 seconds.
Outputs to "kundur_out.lst" and "kundur_out.npz".
Outputs written in 0.0226 seconds.
```

```
True
```

```
ss.dae.ts
```

```
<andes.variables.dae.DAETimeSeries at 0x7f7bc75c1b50>
```

`ss.dae.ts` has four commonly used attributes: `t` for time stamps, `xy` for variables (differential and then algebraic), `z` for discontinuous states, and `df` for the dataframe of all.

- Each point in `ss.dae.ts.t` correspond to a row in `ss.dae.ts.xy`.
- Each column in `ss.dae.ts.xy` correspond to a variable, whose name can be located in `ss.dae.xy_name`, for all timestamps.
- `z` is not stored by default unless one enables it before simulation by setting `ss.TDS.config.store_z = 1`.
- `df` is not built by default but can be manually triggered after simulation by calling `ss.dae.ts.unpack(df=True)`

The following are some statistics of the shapes of arrays:

```
ss.dae.ts.t.shape
```

```
(603, )
```

```
ss.dae.ts.xy.shape # x-axis is for time stamps, and y-axis is for variables
```

```
(603, 201)
```

```
len(ss.dae.xy_name)
```

```
201
```

## Extracting Variable Data

Let's extract the data for rotor speed (variable `omega`) of GENROU generators. The first step to extract variable data is to determine the type of the variable: differential or algebraic. One can print the variable to see the type:

```
ss.GENROU.omega
```

```
State: GENROU.omega, a=[4 5 6 7], v=[1.00165687 1.00166417 1.00182915 1.  
↪00184744], e=[-0.00251535 -0.00375723 0.00131868 0.00182826]
```

The output shows that `omega` is a state (differential variable), which should be looked up in `ss.dae.x`. For algebraic variables such as `ss.Bus.v`, they should be looked up in `ss.dae.y`.

Therefore, all `omega` variables can be extracted as follows:

```
omega = ss.dae.ts.x[:, ss.GENROU.omega.a]
```

where the `:` in the first axis will access such data for all time stamps, and `ss.GENROU.omega.a` stores the addresses of all `omega` into `ss.dae.x`.

To access all bus voltages (algebraic variable `v`) of the generators, one can use:

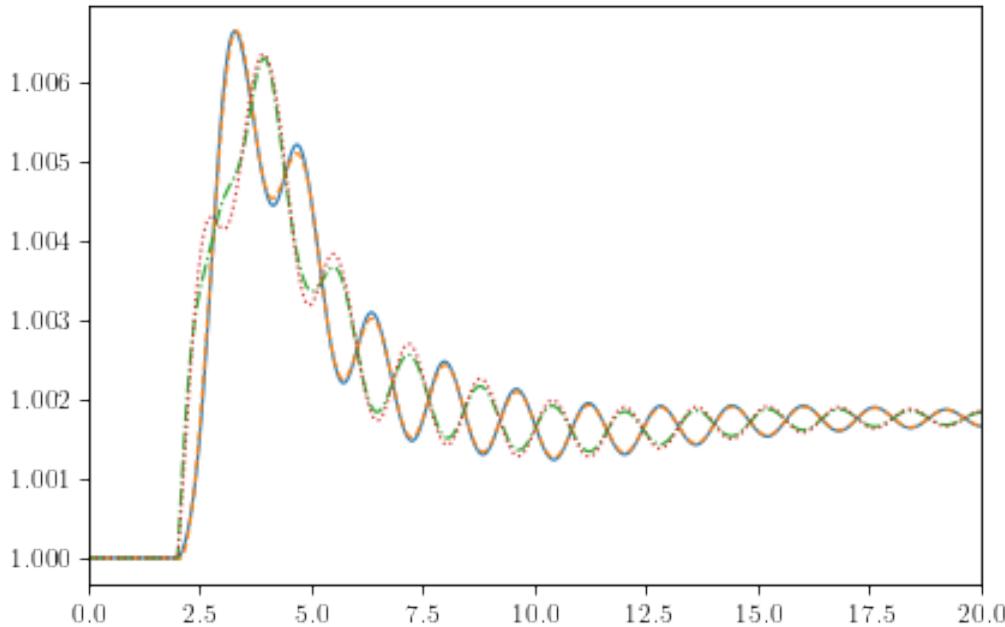
```
ss.dae.ts.y[:, ss.GENROU.v.a]
```

```
array([[1.          , 1.          , 1.          , 1.          , 1.          ],
       [1.          , 1.          , 1.          , 1.          , 1.          ],
       [1.          , 1.          , 1.          , 1.          , 1.          ],
       ...,
       [1.00240968, 1.00148908, 0.99526693, 1.00007159],
       [1.00249935, 1.00159007, 0.99515528, 0.99997846],
       [1.00259067, 1.0016924 , 0.99504062, 0.999883 ]])
```

These data correspond to the timestamps stored in `ss.dae.ts.t`. One can process such data as necessary.

To show verify the extracted data, we plot them with `ss.TDS.plt.plot_data`.

```
ss.TDS.plt.plot_data(ss.dae.ts.t, omega )
```



```
(<Figure size 480x320 with 1 Axes>, <AxesSubplot:>)
```

## 2.2.4 Altering Fault duration

`Alter` can be used for updating any model parameter. We show another example of updating the duration of a fault. Using the `ieee14_fault.xlsx` test case, we have

```
ss = andes.run(get_case("ieee14/ieee14_fault.xlsx"))
```

```
Working directory: "/home/hacui/repos/andes/examples"
> Loaded config from file "/home/hacui/.andes/andes.rc"
> Reloaded generated Python code of module "pycode".
Parsing input file "/home/hacui/repos/andes/andes/cases/ieee14/ieee14_fault.
˓→xlsx"...
```

(continues on next page)

(continued from previous page)

```

Input file parsed in 0.0546 seconds.
System internal structure set up in 0.0348 seconds.
-> System connectivity check results:
    No islanded bus detected.
    System is interconnected.
    Each island has a slack bus correctly defined and enabled.

-> Power flow calculation
    Numba: Off
    Sparse solver: KLU
    Solution method: NR method
Power flow initialized in 0.0037 seconds.
0: |F(x)| = 0.5605182134
1: |F(x)| = 0.006202200332
2: |F(x)| = 5.819382824e-06
3: |F(x)| = 6.96508129e-12
Converged in 4 iterations in 0.0047 seconds.
Initialization for dynamics completed in 0.0353 seconds.
Initialization was successful.
Report saved to "ieee14_fault_out.txt" in 0.0021 seconds.

```

```
-> Single process finished in 0.3141 seconds.
```

Again, if you need to add devices, you should use `ss = andes.load(..., setup=False)` and `ss.setup()` instead of `andes.run()`.

List the existing Fault devices:

```
ss.Fault.as_df()
```

	idx	u	name	bus	tf	tc	xf	rf	
uid	0	1	1.0	Fault_1	9	1.0	1.1	0.0001	0.0

One fault on Bus 9 is applied at t=1.0 sec and cleared at t=1.1 sec. Suppose that we want to clear the fault at t = 1.05 sec, we can do

```
ss.Fault.alter('tc', 1, 1.05) # arguments are `src`, `idx`, `value`
```

where `tc` is the parameter to alter, 1 is the `idx` of the Fault to find, and 1.05 is the new value. Inspect the Fault devices to see the updated value. The simulation for the new system can be performed next.

```
ss.Fault.as_df()
```

	idx	u	name	bus	tf	tc	xf	rf	
uid	0	1	1.0	Fault_1	9	1.0	1.05	0.0001	0.0

## 2.2.5 Cleanup

```
!andes misc -C
```

```
"/home/hacui/repos/andes/examples/kundur_full_out.txt" removed.  
"/home/hacui/repos/andes/examples/kundur_out.1st" removed.  
"/home/hacui/repos/andes/examples/kundur_out.npz" removed.  
"/home/hacui/repos/andes/examples/ieee14_fault_out.txt" removed.  
"/home/hacui/repos/andes/examples/kundur_out.txt" removed.
```

```
!rm new_system.xlsx
```

## 2.3 Inspecting Models

First of all, import `andes` and configure the logger to the `WARNING` level (30).

```
import andes  
andes.config_logger(30)
```

### 2.3.1 Inspect Model Equations

Create an empty `andes.System` object and call `prepare(nomp=True)` to generate equations and pretty print, where `nomp` disables multiprocessing so that equations can be properly returned. This operation may take a moment.

To only generate equations for a specific model, such as `GENCLS`, do:

```
ss = andes.System()  
ss.GENCLS.prepare()
```

### List all models

```
print(ss.supported_models())
```

Supported Groups and Models

Group		Models
ACLine		Line
ACShort		Jumper
ACTopology		Bus
Calculation		ACE, ACEc, COI
Collection		Area
DCLink		Ground, R, L, C, RCp, RCs, RLS, RLCs, RLCP

(continues on next page)

(continued from previous page)

DCTopology	Node
DG	PVD1, ESD1, EV1, EV2
DGProtection	DGPRCT1, DGPRCTExt
DynLoad	ZIP, FLoad
Exciter	EXDC2, IEEEX1, ESDC2A, EXST1, ESST3A, SEXS, IEEET1,   EXAC1, EXAC4, ESST4B, AC8B, IEEET3, ESAC1A, ESST1A
FreqMeasurement	BusFreq, BusROCOF
Information	Summary
Motor	Motor3, Motor5
PLL	PLL1
PSS	IEEEST, ST2CUT
PhasorMeasurement	PMU
RenAerodynamics	WTARA1, WTARV1
RenExciter	REECA1, REECA1E, REECA1G
RenGen	REGCA1, REGCP1, REGCV1, REGCV2
RenGovernor	WTDIA1, WTDS
RenPitch	WTPTA1
RenPlant	REPCA1
RenTorque	WTTQA1
StaticACDC	VSCShunt
StaticGen	PV, Slack
StaticLoad	PQ
StaticShunt	Shunt, ShuntTD, ShuntSw
SynGen	GENCLS, GENROU, PLBFU1, GENROUOS
TimedEvent	Toggle, Fault, Alter
TurbineGov	TG2, TGOV1, TGOV1DB, TGOV1N, TGOV1NDB, IEEEG1, IEESGO,   GAST, HYGOV, HYGOVDB
Undefined	TimeSeries
VoltComp	IEEEVC

### 2.3.2 Check model documentation

To check the documentation for the model, print the return of `doc()` for the model instance.

For example, the documentation for `GENCLS` can be printed with

```
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

(continues on next page)

(continued from previous page)

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,power	
	(2H)				
ra	armature resistance	0		z	
xl	leakage reactance	0		z	
xd1	d-axis transient reactance	0.302		z	
kp	active power feedback gain	0			
	gain				
kw	speed feedback gain	0			
S10	first saturation factor	0			
S12	second saturation factor	1			
gammap	P ratio of linked static gen	1			
gammaq	Q ratio of linked static gen	1			
subidx	Generator idx in plant; only used by PSS/E data	0			

## 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 current	p.u (kV)	v_str
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

(continues on next page)

(continued from previous page)

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	u * (2 * pi * fn) * (omega - 1)	
omega	State	u * (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	u * v * sin(delta - a) - vd
vq	Algeb	u * v * cos(delta - a) - vq
tm	Algeb	tm0 - tm
te	Algeb	u * (psid * Iq - psiq * Id) - te
vf	Algeb	u * vf0 - vf
XadIfd	Algeb	u * vf0 - XadIfd
Pe	Algeb	u * (vd * Id + vq * Iq) - Pe
Qe	Algeb	u * (vq * Id - vd * Iq) - Qe
psid	Algeb	u * (ra*Iq + vq) - psid
psiq	Algeb	u * (ra*Id + vd) + psiq
a	ExtAlgeb	-u * (vd * Id + vq * Iq)
v	ExtAlgeb	-u * (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

(continues on next page)

(continued from previous page)

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

Config Fields in [GENCLS]

Option	Value	Info	Acceptable 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	

## Pretty print of variables

All symbols are stored in the attributes of Model.syms. For example,

```
ss.GENCLS.syms.xy
```

$$\begin{bmatrix} \delta \\ \omega \\ I_d \\ I_q \\ V_d \\ V_q \\ \tau_m \\ \tau_e \\ v_f \\ X_{ad} \\ I_{fd} \\ P_e \\ Q_e \\ \psi_d \\ \psi_q \\ \theta \\ V \end{bmatrix}$$

Differential variables comes before algebraic variables.

```
ss.GENCLS.states
```

```
OrderedDict([('delta', State: GENCLS.delta, []),
             ('omega', State: GENCLS.omega, [])])
```

```
ss.GENCLS.algebs
```

```
OrderedDict([('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, [])])
```

## Pretty print of equations

Formatted equations are stored in each model. The following attributes of `Model.syms` are available for equation printing.

- `f`: differential equations
- `g`: algebraic equations
- `df`:  $df/dx$
- `dg`:  $dg/dx$

```
ss.GENCLS.syms.f
```

$$\begin{bmatrix} 2\pi f u (\omega - 1) \\ u (-D(\omega - 1) - \tau_e + \tau_m) \end{bmatrix}$$

```
ss.GENCLS.syms.g
```

$$\begin{bmatrix} I_d x q + \psi_d - v_f \\ I_q x q + \psi_q \\ V u \sin(\delta - \theta) - V_d \\ V u \cos(\delta - \theta) - V_q \\ -\tau_m + \tau_{m0} \\ -\tau_e + u(-I_d \psi_q + I_q \psi_d) \\ u v_{f0} - v_f \\ -X_{ad} I_{fd} + u v_{f0} \\ -P_e + u(I_d V_d + I_q V_q) \\ -Q_e + u(I_d V_q - I_q V_d) \\ -\psi_d + u(I_q r_a + V_q) \\ \psi_q + u(I_d r_a + V_d) \\ -u(I_d V_d + I_q V_q) \\ -u(I_d V_q - I_q V_d) \end{bmatrix}$$

ss.GENCLS.syms.df

$$\begin{bmatrix} 0 & 2\pi f u & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -D u & 0 & 0 & 0 & 0 & u & -u & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

ss.GENCLS.syms.dg

$$\begin{bmatrix} 0 & 0 & xq & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & xq & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ V u \cos(\delta - \theta) & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -V u \cos(\delta - \theta) \\ -V u \sin(\delta - \theta) & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & V u \sin(\delta - \theta) \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\psi_q u & \psi_d u & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & I_q u & -I_d u & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & V_d u & V_q u & I_d u & I_q u & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & V_q u & -V_d u & -I_q u & I_d u & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & r_a u & 0 & u & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & r_a u & 0 & u & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & -V_d u & -V_q u & -I_d u & -I_q u & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -V_q u & V_d u & I_q u & -I_d u & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

## Pretty print of services

The list of services is in `services`.

```
ss.GENCLS.services
```

```
OrderedDict([('p0', ConstService: GENCLS.p0, v=0.0),
            ('q0', ConstService: GENCLS.q0, v=0.0),
            ('_V', ConstService: GENCLS._V, v=0.0),
            ('_S', ConstService: GENCLS._S, v=0.0),
            ('_I', ConstService: GENCLS._I, v=0.0),
            ('_E', ConstService: GENCLS._E, v=0.0),
            ('_deltac', ConstService: GENCLS._deltac, v=0.0),
            ('delta0', ConstService: GENCLS.delta0, v=0.0),
            ('vdq', ConstService: GENCLS.vdq, v=0.0),
            ('Idq', ConstService: GENCLS.Idq, v=0.0),
            ('Id0', ConstService: GENCLS.Id0, v=0.0),
            ('Iq0', ConstService: GENCLS.Iq0, v=0.0),
            ('vd0', ConstService: GENCLS.vd0, v=0.0),
            ('vq0', ConstService: GENCLS.vq0, v=0.0),
            ('tm0', ConstService: GENCLS.tm0, v=0.0),
            ('psid0', ConstService: GENCLS.psid0, v=0.0),
            ('psiq0', ConstService: GENCLS.psiq0, v=0.0),
            ('vf0', ConstService: GENCLS.vf0, v=0.0)])
```

Service equations are in `Model.syms.s`. For example, services of `GENCLS` is in

```
ss.GENCLS.syms.s
```

```
[P_{0s}*\gamma_P,
Q_{0s}*\gamma_Q,
V*exp(I*\theta),
P_0 - I*Q_0,
S/conj(V_c),
I_c*(r_a + I*xq) + V_c,
log(E/Abs(E)),
u*im(\delta_c),
V_c*u*exp(-\delta_c + 0.5*I*pi),
I_c*u*exp(-\delta_c + 0.5*I*pi),
re(I_{dq}),
im(I_{dq}),
re(V_{dq}),
im(V_{dq}),
u*(I_{d0}*(I_{d0}*r_a + V_{d0}) + I_{q0}*(I_{q0}*r_a + V_{q0})),
I_{q0}*r_a*u + V_{q0},
-I_{d0}*r_a*u - V_{d0},
I_{d0}*xq + I_{q0}*r_a + V_{q0}]
```

## 2.4 Eigenvalue Analysis

### 2.4.1 Run Eigenvalue Analysis

```
import numpy as np

import andes
from andes.utils.paths import list_cases, get_case

case_path = get_case('kundur/kundur_full.xlsx')
```

Pass the routine name EIG or eig to andes.run.

```
ss = andes.run(case_path, routine='eig')
```

```
Working directory: "/home/hacui/repos/andes/examples"
> Loaded config from file "/home/hacui/.andes/andes.rc"
> Loaded generated Python code in "/home/hacui/.andes/pycode".
Parsing input file "/home/hacui/repos/andes/andes/cases/kundur/kundur_full.xlsx"...
Input file parsed in 0.1985 seconds.
System internal structure set up in 0.0309 seconds.
-> System connectivity check results:
  No islanded bus detected.
  System is interconnected.
  Each island has a slack bus correctly defined and enabled.

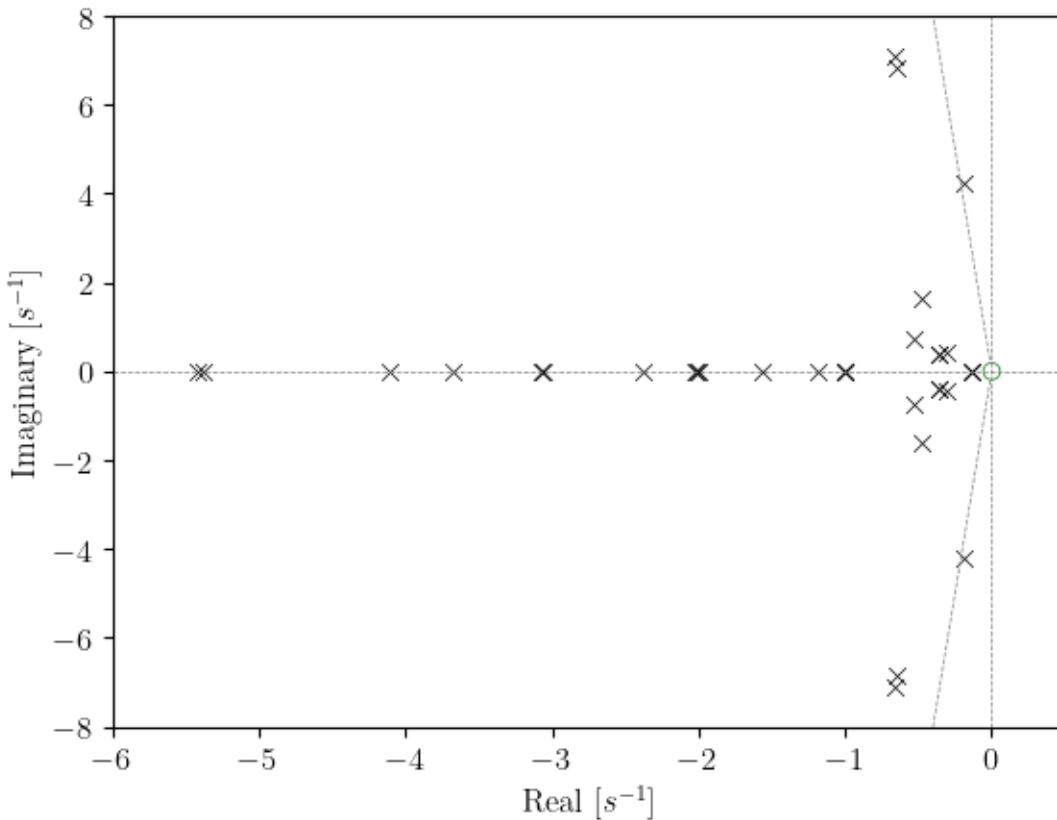
-> Power flow calculation
    Numba: Off
    Sparse solver: KLU
    Solution method: NR method
Power flow initialized in 0.0042 seconds.
0: |F(x)| = 14.9282832
1: |F(x)| = 3.608627841
2: |F(x)| = 0.1701107882
3: |F(x)| = 0.002038626956
4: |F(x)| = 3.745104027e-07
Converged in 5 iterations in 0.0070 seconds.
Report saved to "kundur_full_out.txt" in 0.0010 seconds.
Initialization for dynamics completed in 0.0301 seconds.
Initialization was successful.

-> Eigenvalue Analysis:
  Positive      0
  Zeros         1
  Negative     52
Eigenvalue analysis finished in 0.0051 seconds.
Report saved to "kundur_full_eig.txt".
```

```
-> Single process finished in 0.5363 seconds.
```

## 2.4.2 Plotting in S-Domain

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



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

## 2.4.3 Report

Report is saved to the file and can be loaded into the notebook.

```
with open('kundur_full_eig.txt', 'r') as f:
    print(f.read())
```

```
ANDES 1.7.5.post2.dev0+ga53a31fa
Copyright (C) 2015-2022 Hantao Cui

ANDES comes with ABSOLUTELY NO WARRANTY
Case file: /home/hacui/repos/andes/andes/cases/kundur/kundur_full.xlsx
Report time: 07/05/2022 08:57:50 PM

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

(continues on next page)

(continued from previous page)

## EIGENVALUE ANALYSIS REPORT

Positives	0
Zeros	1
Negatives	52

## STATISTICS

↔Damped Freq.	Most Associated Frequency	Real		Imag.
		Damping	[%]	
#1 ↔ 0	LL_x EXDC2 1 0	-1	0	0
#2 ↔ 0	LL_x EXDC2 2 0	-1	0	0
#3 ↔ 0	LL_x EXDC2 3 0	-1	0	0
#4 ↔ 0	LL_x EXDC2 4 0	-1	0	0
#5 ↔ 0.10223	LS_y EXDC2 3 7.8296	-49.191 99.991	0.64233	0
#6 ↔ 0.10223	LS_y EXDC2 3 7.8296	-49.191 99.991	-0.64233	0
#7 ↔ 0	LS_y EXDC2 1 0	-49.535 0	0	0
#8 ↔ 0	LS_y EXDC2 4 0	-49.515 0	0	0
#9 ↔ 0.062627	LA_y EXDC2 2 7.831	-49.202 99.997	0.39349	0
#10 ↔ 0.062627	LA_y EXDC2 2 7.831	-49.202 99.997	-0.39349	0
#11 ↔ 0	LA_y EXDC2 1 0	-48.885 0	0	0
#12 ↔ 0	LA_y EXDC2 4 0	-48.905 0	0	0
#13 ↔ 0	e2d GENROU 2 0	-36.896 0	0	0
#14 ↔ 0	e2d GENROU 4 0	-36.75 0	0	0
#15 ↔ 0	e2d GENROU 1 0	-34.732 0	0	0
#16 ↔ 0	e2q GENROU 3 0	-33.89 0	0	0
#17 ↔ 0	e2q GENROU 2 0	-33.526 0	0	0
#18 ↔ 0	e2d GENROU 3 0	-31.242 0	0	0
#19 ↔ 0	e2q GENROU 4 0	-27.613 0	0	0
#20 ↔ 0	e2q GENROU 1 0	-24.954 0	0	0

(continues on next page)

(continued from previous page)

#21		delta GENROU 3	-0.6564	7.086	[ ]
↔	1.1278	1.1326	9.224		[ ]
#22		delta GENROU 3	-0.6564	-7.086	[ ]
↔	1.1278	1.1326	9.224		[ ]
#23		delta GENROU 2	-0.65252	6.8343	[ ]
↔	1.0877	1.0927	9.5046		[ ]
#24		delta GENROU 2	-0.65252	-6.8343	[ ]
↔	1.0877	1.0927	9.5046		[ ]
#25		delta GENROU 1	-0.19177	4.2247	[ ]
↔	0.67239	0.67308	4.5346		[ ]
#26		delta GENROU 1	-0.19177	-4.2247	[ ]
↔	0.67239	0.67308	4.5346		[ ]
#27		e1d GENROU 2	-5.4189	0	[ ]
↔	0	0	0		[ ]
#28		e1d GENROU 3	-5.3729	0	[ ]
↔	0	0	0		[ ]
#29		e1d GENROU 4	-4.1119	0	[ ]
↔	0	0	0		[ ]
#30		e1d GENROU 2	-3.6722	0	[ ]
↔	0	0	0		[ ]
#31		vp EXDC2 2	-0.48245	1.628	[ ]
↔	0.25911	0.27025	28.413		[ ]
#32		vp EXDC2 2	-0.48245	-1.628	[ ]
↔	0.25911	0.27025	28.413		[ ]
#33		vp EXDC2 4	-3.0736	0	[ ]
↔	0	0	0		[ ]
#34		vp EXDC2 1	-3.0635	0	[ ]
↔	0	0	0		[ ]
#35		vp EXDC2 1	-2.3787	0	[ ]
↔	0	0	0		[ ]
#36		LAG_y TGOV1 1	-1.5702	0	[ ]
↔	0	0	0		[ ]
#37		LAG_y TGOV1 4	-1.9922	0	[ ]
↔	0	0	0		[ ]
#38		LAG_y TGOV1 2	-2.0109	0	[ ]
↔	0	0	0		[ ]
#39		LAG_y TGOV1 3	-2.0194	0	[ ]
↔	0	0	0		[ ]
#40		W_x EXDC2 2	-1.1816	0	[ ]
↔	0	0	0		[ ]
#41		W_x EXDC2 4	-0.53573	0.73459	[ ]
↔	0.11691	0.1447	58.924		[ ]
#42		W_x EXDC2 4	-0.53573	-0.73459	[ ]
↔	0.11691	0.1447	58.924		[ ]
#43*		delta GENROU 3	-1.1783e-14	0	[ ]
↔	0	0	0		[ ]
#44		LL_x TGOV1 2	-0.30739	0.44664	[ ]
↔	0.071085	0.086293	56.693		[ ]
#45		LL_x TGOV1 2	-0.30739	-0.44664	[ ]
↔	0.071085	0.086293	56.693		[ ]
#46		W_x EXDC2 1	-0.36003	0.38874	[ ]
↔	0.061869	0.084328	67.95		[ ]

(continues on next page)

(continued from previous page)

#47		W_x EXDC2 1	-0.36003	-0.38874	[ ]
↔	0.061869	0.084328	67.95		[ ]
#48		W_x EXDC2 3	-0.35993	0.37956	[ ]
↔	0.060409	0.083251	68.81		[ ]
#49		W_x EXDC2 3	-0.35993	-0.37956	[ ]
↔	0.060409	0.083251	68.81		[ ]
#50		LL_x TGOV1 1	-0.14157	0	[ ]
↔	0	0	0		[ ]
#51		LL_x TGOV1 2	-0.14203	0	[ ]
↔	0	0	0		[ ]
#52		LL_x TGOV1 3	-0.14202	0	[ ]
↔	0	0	0		[ ]
EIGENVALUE DATA					
		#1	#2	#3	[ ]
↔	#4	#5	#6	#7	[ ]
delta GENROU 1		0	0	0	[ ]
↔	0	0	0	4e-05	[ ]
delta GENROU 2		0	0	0	[ ]
↔	0	0	0	6e-05	[ ]
delta GENROU 3		0	0	0	[ ]
↔	0	0	0	2e-05	[ ]
delta GENROU 4		0	0	0	[ ]
↔	0	0	0	1e-05	[ ]
omega GENROU 1		0	0	0	[ ]
↔	0	0	0	2e-05	[ ]
omega GENROU 2		0	0	0	[ ]
↔	0	0	0	3e-05	[ ]
omega GENROU 3		0	0	0	[ ]
↔	0	0	0	1e-05	[ ]
omega GENROU 4		0	0	0	[ ]
↔	0	0	0	0	[ ]
e1q GENROU 1		0	0	0	[ ]
↔	0	0.00084	0.00084	0.00306	[ ]
e1q GENROU 2		0	0	0	[ ]
↔	0	0.0019	0.0019	0.00224	[ ]
e1q GENROU 3		0	0	0	[ ]
↔	0	0.0026	0.0026	0.00069	[ ]
e1q GENROU 4		0	0	0	[ ]
↔	0	0.00118	0.00118	0.00063	[ ]
e1d GENROU 1		0	0	0	[ ]
↔	0	0	0	0	[ ]
e1d GENROU 2		0	0	0	[ ]
↔	0	0	0	0	[ ]
e1d GENROU 3		0	0	0	[ ]
↔	0	0	0	0	[ ]
e1d GENROU 4		0	0	0	[ ]
↔	0	0	0	0	[ ]
e2d GENROU 1		0	0	0	[ ]
↔	0	0.00839	0.00839	0.02012	[ ]
e2d GENROU 2		0	0	0	[ ]

(continues on next page)

(continued from previous page)

↔	0	0.01824	0.01824	0.01364		
e2d GENROU	3		0	0	0	0
↔	0	0.02784	0.02784	0.00467		
e2d GENROU	4		0	0	0	0
↔	0	0.01361	0.01361	0.00474		
e2q GENROU	1		0	0	0	0
↔	0	0.00096	0.00096	0.00036		
e2q GENROU	2		0	0	0	0
↔	0	0.00026	0.00026	0.00114		
e2q GENROU	3		0	0	0	0
↔	0	3e-05	3e-05	0.00031		
e2q GENROU	4		0	0	0	0
↔	0	0.00118	0.00118	9e-05		
LAG_y TGOV1	1		0	0	0	0
↔	0	0	0	0	0	0
LAG_y TGOV1	2		0	0	0	0
↔	0	0	0	0	0	0
LAG_y TGOV1	3		0	0	0	0
↔	0	0	0	0	0	0
LAG_y TGOV1	4		0	0	0	0
↔	0	0	0	0	0	0
LL_x TGOV1	1		0	0	0	0
↔	0	0	0	0	0	0
LL_x TGOV1	2		0	0	0	0
↔	0	0	0	0	0	0
LL_x TGOV1	3		0	0	0	0
↔	0	0	0	0	0	0
LL_x TGOV1	4		0	0	0	0
↔	0	0	0	0	0	0
vp EXDC2	1		0	0	0	0
↔	0	0.00157	0.00157	0.00231		
vp EXDC2	2		0	0	0	0
↔	0	0.00352	0.00352	0.00168		
vp EXDC2	3		0	0	0	0
↔	0	0.00346	0.00346	0.00037		
vp EXDC2	4		0	0	0	0
↔	0	0.00202	0.00202	0.00043		
LS_y EXDC2	1		0	0	0	0
↔	0	0.06526	0.06526	0.53032		
LS_y EXDC2	2		0	0	0	0
↔	0	0.14386	0.14386	0.37893		
LS_y EXDC2	3		0	0	0	0
↔	0	0.30715	0.30715	0.18162		
LS_y EXDC2	4		0	0	0	0
↔	0	0.08575	0.08575	0.10168		
LL_x EXDC2	1	0.66046		0	0	0
↔	0	0	0	0	0	0
LL_x EXDC2	2		0.66046		0	0
↔	0	0	0	0	0	0
LL_x EXDC2	3		0	0	1	0
↔	0	0	0	0	0	0
LL_x EXDC2	4		0	0	0	0

(continues on next page)

(continued from previous page)

↔ 0.65832	0	0	0	0
LA_Y EXDC2 1	0	0.06704	0.06704	0.22072
↔ 0	0.06704	0	0	0
LA_Y EXDC2 2	0	0	0	0
↔ 0	0.20271	0.20271	0.21636	0
LA_Y EXDC2 3	0	0	0	0
↔ 0	0.30051	0.30051	0.072	0
LA_Y EXDC2 4	0	0	0	0
↔ 0	0.1453	0.1453	0.06982	0
W_X EXDC2 1	0	5e-05	5e-05	0.00015
↔ 0	5e-05	0	0	0
W_X EXDC2 2	0	0	0	0
↔ 0	0.00012	0.00012	0.00013	0
W_X EXDC2 3	0	0	0	0
↔ 0	0.00017	0.00017	4e-05	0
W_X EXDC2 4	0	0	0	0
↔ 0	8e-05	8e-05	4e-05	0
PARTICIPATION FACTORS [1/8]				
	#11	#12	#13	#14
		#8	#9	#10
↔				
delta GENROU 1		2e-05	1e-05	1e-05
↔ 5e-05		2e-05	0.00044	0.00047
delta GENROU 2		1e-05	0	0
↔ 6e-05		1e-05	0.00078	0.00049
delta GENROU 3		5e-05	0	0
↔ 2e-05		6e-05	0.00062	0.00061
delta GENROU 4		5e-05	1e-05	1e-05
↔ 1e-05		6e-05	0.00032	0.00067
omega GENROU 1		1e-05	1e-05	1e-05
↔ 3e-05		1e-05	0.00025	0.00027
omega GENROU 2		1e-05	0	0
↔ 4e-05		1e-05	0.00045	0.00028
omega GENROU 3		2e-05	0	0
↔ 1e-05		2e-05	0.00025	0.00024
omega GENROU 4		2e-05	0	0
↔ 0		2e-05	0.00012	0.00025
e1q GENROU 1		0.00117	0.00225	0.00225
↔ 0.0031		0.00115	0.0003	0.00033
e1q GENROU 2		0.00043	0.00285	0.00285
↔ 0.00224		0.00041	0.00042	0.00021
e1q GENROU 3		0.00184	0.00167	0.00167
↔ 0.00067		0.00185	0.00028	0.00027
e1q GENROU 4		0.00329	0.00134	0.00134
↔ 0.00061		0.00333	0.00019	0.00041
e1d GENROU 1		0	0	0
↔ 0		0	2e-05	1e-05
e1d GENROU 2		0	0	0
↔ 0		0	7e-05	2e-05
e1d GENROU 3		0	0	0
↔ 0		0	5e-05	5e-05

(continues on next page)

(continued from previous page)

e1d GENROU 4	0	0	0	0
↔ 0	0	2e-05	1e-05	0
e2d GENROU 1	0.00784	0.01923	0.01923	0
↔ 0.02323	0.00867	0.23812	0.24595	0
e2d GENROU 2	0.00261	0.02337	0.02337	0
↔ 0.01559	0.00284	0.33371	0.16432	0
e2d GENROU 3	0.01258	0.01507	0.01507	0
↔ 0.00518	0.01432	0.2465	0.23261	0
e2d GENROU 4	0.02533	0.01309	0.01309	0
↔ 0.00526	0.02886	0.17228	0.34779	0
e2q GENROU 1	0.00015	0.0011	0.0011	0
↔ 0.00039	0.00016	0.00111	0.00037	0
e2q GENROU 2	0.00038	0.00045	0.00045	0
↔ 0.00121	0.00039	0.0033	0.00106	0
e2q GENROU 3	0.00116	7e-05	7e-05	0
↔ 0.00032	0.00123	0.00241	0.00234	0
e2q GENROU 4	0.00041	0.00079	0.00079	0
↔ 9e-05	0.00044	0.00069	0.00043	0
LAG_y TGOV1 1	0	0	0	0
↔ 0	0	0	0	0
LAG_y TGOV1 2	0	0	0	0
↔ 0	0	0	0	0
LAG_y TGOV1 3	0	0	0	0
↔ 0	0	0	0	0
LAG_y TGOV1 4	0	0	0	0
↔ 0	0	0	0	0
LL_x TGOV1 1	0	0	0	0
↔ 0	0	0	0	0
LL_x TGOV1 2	0	0	0	0
↔ 0	0	0	0	0
LL_x TGOV1 3	0	0	0	0
↔ 0	0	0	0	0
LL_x TGOV1 4	0	0	0	0
↔ 0	0	0	0	0
vp EXDC2 1	0.00094	0.00415	0.00415	0
↔ 0.01364	0.00476	0.00032	0.00034	0
vp EXDC2 2	0.00034	0.00522	0.00522	0
↔ 0.00979	0.00169	0.00042	0.0002	0
vp EXDC2 3	0.00105	0.0022	0.0022	0
↔ 0.0021	0.00546	0.0002	0.00019	0
vp EXDC2 4	0.00242	0.00226	0.00226	0
↔ 0.00246	0.01263	0.00018	0.00038	0
LS_y EXDC2 1	0.19482	0.203	0.203	0
↔ 0.22143	0.08347	0.00069	0.00074	0
LS_y EXDC2 2	0.06924	0.25052	0.25052	0
↔ 0.15607	0.02909	0.0009	0.00042	0
LS_y EXDC2 3	0.46474	0.22878	0.22878	0
↔ 0.07262	0.2043	0.00092	0.00088	0
LS_y EXDC2 4	0.51255	0.11281	0.11281	0
↔ 0.04075	0.22639	0.00041	0.00085	0
LL_x EXDC2 1	0	0	0	0
↔ 0	0	0	0	0

(continues on next page)

(continued from previous page)

LL_x EXDC2 2	0	0	0	0	0
LL_x EXDC2 3	0	0	0	0	0
LL_x EXDC2 4	0	0	0	0	0
LA_y EXDC2 1	0.08621	0.2062	0.2062	0.2062	0
LA_y EXDC2 2	0.19058	0.00079	0.00084	0.00084	0
LA_y EXDC2 3	0.04203	0.34908	0.34908	0.34908	0
LA_y EXDC2 4	0.09112	0.00141	0.00066	0.00066	0
LA_y EXDC2 5	0.19588	0.22134	0.22134	0.22134	0
LA_y EXDC2 6	0.44428	0.001	0.00095	0.00095	0
LA_y EXDC2 7	0.37416	0.18904	0.18904	0.18904	0
LA_y EXDC2 8	0.85269	0.00077	0.0016	0.0016	0
W_x EXDC2 1	6e-05	0.00014	0.00014	0.00014	0
W_x EXDC2 2	0.00014	0	0	0	0
W_x EXDC2 3	2e-05	0.00021	0.00021	0.00021	0
W_x EXDC2 4	5e-05	0	0	0	0
W_x EXDC2 5	0.00011	0.00013	0.00013	0.00013	0
W_x EXDC2 6	0.00026	0	0	0	0
W_x EXDC2 7	0.00021	0.00011	0.00011	0.00011	0
W_x EXDC2 8	0.0001	0	0	0	0

## PARTICIPATION FACTORS [2/8]

	#15	#16	#17	
↔	#18	#19	#20	#21
delta GENROU 1	0.0013	0.00058	0.00086	0
↔ 4e-05	1e-05	0.00014	0.04024	0
delta GENROU 2	0.00034	0.00024	0.00268	0
↔ 6e-05	7e-05	2e-05	0.08541	0
delta GENROU 3	0.00011	0.00276	0.00076	0
↔ 5e-05	5e-05	0.0002	0.23816	0
delta GENROU 4	0.00079	0.00199	1e-05	0
↔ 0.00011	1e-05	1e-05	0.16086	0
omega GENROU 1	0.00075	0.00034	0.0005	0
↔ 3e-05	0	8e-05	0.02377	0
omega GENROU 2	0.0002	0.00014	0.00155	0
↔ 3e-05	4e-05	1e-05	0.05045	0
omega GENROU 3	4e-05	0.00109	0.0003	0
↔ 2e-05	2e-05	8e-05	0.0965	0
omega GENROU 4	0.00029	0.00074	1e-05	0
↔ 4e-05	0	0	0.06107	0
e1q GENROU 1	0.00054	6e-05	0.00042	0
↔ 0.00094	4e-05	0.0003	0.00067	0
e1q GENROU 2	0.00065	0.00022	2e-05	0
↔ 0.00115	0.00016	0.00097	0.00141	0
e1q GENROU 3	0.00038	0.00019	0.00019	0
↔ 0.00144	0.00011	0.00123	0.00194	0
e1q GENROU 4	0.00034	1e-05	0.0003	0
↔ 0.00112	2e-05	0.00045	0.0034	0
e1d GENROU 1	0.00197	0.00266	0.00808	0

(continues on next page)

(continued from previous page)

↪ 1e-05	0.02906	0.04674	0.00263	
e1d GENROU 2	0.00033	0.00186	0.01412	↪ ↪
↪ 0.00441	0.01327	0.03129	0.00637	
e1d GENROU 3	0.00138	0.01461	0.00314	↪ ↪
↪ 0.00479	0.0177	0.02276	0.02746	
e1d GENROU 4	0.00094	0.01041	0.00046	↪ ↪
↪ 5e-05	0.03673	0.03898	0.00549	
e2d GENROU 1	0.25874	0.03867	0.05142	↪ ↪
↪ 0.14321	0.00031	0.0033	0.00066	
e2d GENROU 2	0.16397	0.03747	0.0468	↪ ↪
↪ 0.20085	0.00352	0.02285	0.00139	
e2d GENROU 3	0.11321	0.0137	0.07851	↪ ↪
↪ 0.27348	0.00187	0.03733	0.00213	
e2d GENROU 4	0.18053	0.00845	0.08023	↪ ↪
↪ 0.18763	0.00026	0.00906	0.00371	
e2q GENROU 1	0.05939	0.06822	0.19355	↪ ↪
↪ 0.00017	0.23112	0.22378	0.00134	
e2q GENROU 2	0.09922	0.04763	0.3366	↪ ↪
↪ 0.06848	0.10503	0.1491	0.00325	
e2q GENROU 3	0.04116	0.36911	0.07409	↪ ↪
↪ 0.07356	0.13858	0.10724	0.01383	
e2q GENROU 4	0.028	0.26219	0.0109	↪ ↪
↪ 0.00082	0.28669	0.18311	0.00276	
LAG_y TGOV1 1	0	0	0	
↪ 0	0	0	0.00067	↪ ↪
LAG_y TGOV1 2	0	0	0	
↪ 0	0	0	0.00142	↪ ↪
LAG_y TGOV1 3	0	0	0	
↪ 0	0	0	0.00355	↪ ↪
LAG_y TGOV1 4	0	0	0	
↪ 0	0	0	0.00239	↪ ↪
LL_x TGOV1 1	0	0	0	
↪ 0	0	0	3e-05	↪ ↪
LL_x TGOV1 2	0	0	0	
↪ 0	0	0	7e-05	↪ ↪
LL_x TGOV1 3	0	0	0	
↪ 0	0	0	0.00014	↪ ↪
LL_x TGOV1 4	0	0	0	
↪ 0	0	0	9e-05	↪ ↪
vp EXDC2 1	0.00033	3e-05	0.00062	
↪ 0.00077	9e-05	0.00049	6e-05	
vp EXDC2 2	0.00084	0.0003	0.00017	↪ ↪
↪ 0.00073	0.00025	0.00122	0.00018	
vp EXDC2 3	0.00034	0.00023	6e-05	
↪ 0.00069	0.00014	0.00101	0.00069	
vp EXDC2 4	0.00022	4e-05	0.00029	
↪ 0.0009	3e-05	0.00061	0.00083	
LS_y EXDC2 1	0.0006	5e-05	0.00101	
↪ 0.00103	9e-05	0.00039	1e-05	
LS_y EXDC2 2	0.00149	0.00049	0.00026	↪ ↪
↪ 0.00095	0.00024	0.00095	3e-05	
LS_y EXDC2 3	0.00129	0.00084	0.0002	↪ ↪

(continues on next page)

(continued from previous page)

↔ 0.00197	0.0003	0.00172	0.00021	
LS_y EXDC2 4	0.00041	6e-05	0.00048	↔ [ ]
↔ 0.00123	3e-05	0.00049	0.00012	
LL_x EXDC2 1	0	0	0	↔ [ ]
↔ 0	0	0	0	↔ [ ]
LL_x EXDC2 2	0	0	0	↔ [ ]
↔ 0	0	0	0	↔ [ ]
LL_x EXDC2 3	0	0	0	↔ [ ]
↔ 0	0	0	0	↔ [ ]
LL_x EXDC2 4	0	0	0	↔ [ ]
↔ 0	0	0	0	↔ [ ]
LA_y EXDC2 1	0.00067	6e-05	0.00113	↔ [ ]
↔ 0.00114	0.0001	0.00042	1e-05	
LA_y EXDC2 2	0.00229	0.00074	0.0004	↔ [ ]
↔ 0.00144	0.00037	0.00143	3e-05	
LA_y EXDC2 3	0.00137	0.00089	0.00021	↔ [ ]
↔ 0.00207	0.00031	0.00179	0.00018	
LA_y EXDC2 4	0.00076	0.00011	0.00089	↔ [ ]
↔ 0.00224	6e-05	0.00089	0.00018	
W_x EXDC2 1	0	0	0	↔ [ ]
↔ 0	0	0	0	↔ [ ]
W_x EXDC2 2	0	0	0	↔ [ ]
↔ 0	0	1e-05	1e-05	
W_x EXDC2 3	0	0	0	↔ [ ]
↔ 0	0	1e-05	3e-05	
W_x EXDC2 4	0	0	0	↔ [ ]
↔ 1e-05	0	0	3e-05	

## PARTICIPATION FACTORS [3/8]

	#22	#23	#24	
↔ #25	#26	#27	#28	
delta GENROU 1	0.04024	0.17926	0.17926	↔ [ ]
↔ 0.17245	0.17245	0.01877	0.00547	
delta GENROU 2	0.08541	0.21636	0.21636	↔ [ ]
↔ 0.08005	0.08005	0.02617	0.00332	
delta GENROU 3	0.23816	0.04918	0.04918	↔ [ ]
↔ 0.09817	0.09817	0.00681	0.02482	
delta GENROU 4	0.16086	0.08069	0.08069	↔ [ ]
↔ 0.14333	0.14333	0.00315	0.02115	
omega GENROU 1	0.02377	0.10604	0.10604	↔ [ ]
↔ 0.10491	0.10491	0.01037	0.00302	
omega GENROU 2	0.05045	0.12794	0.12794	↔ [ ]
↔ 0.04872	0.04872	0.01445	0.00183	
omega GENROU 3	0.0965	0.01995	0.01995	↔ [ ]
↔ 0.04102	0.04102	0.00257	0.00936	
omega GENROU 4	0.06107	0.03068	0.03068	↔ [ ]
↔ 0.05611	0.05611	0.00111	0.00747	
e1q GENROU 1	0.00067	0.00511	0.00511	↔ [ ]
↔ 0.00596	0.00596	0.00153	0.00043	
e1q GENROU 2	0.00141	0.00145	0.00145	↔ [ ]
↔ 0.00491	0.00491	0.00172	0.00027	

(continues on next page)

(continued from previous page)

e1q GENROU 3	0.00194	0.00018	0.00018	[ ]
↔ 0.00041	0.00041	0.00039	0.0016	[ ]
e1q GENROU 4	0.00034	0.00215	0.00215	[ ]
↔ 0.00091	0.00091	0.00022	0.00155	[ ]
e1d GENROU 1	0.00263	0.00462	0.00462	[ ]
↔ 0.00044	0.00044	0.36152	0.09703	[ ]
e1d GENROU 2	0.00637	0.02796	0.02796	[ ]
↔ 0.00049	0.00049	0.49924	0.07963	[ ]
e1d GENROU 3	0.02746	0.00894	0.00894	[ ]
↔ 0.00335	0.00335	0.12193	0.48105	[ ]
e1d GENROU 4	0.00549	0.00157	0.00157	[ ]
↔ 0.00281	0.00281	0.05818	0.38742	[ ]
e2d GENROU 1	0.00066	0.0043	0.0043	[ ]
↔ 0.00204	0.00204	0.00033	8e-05	[ ]
e2d GENROU 2	0.00139	0.00146	0.00146	[ ]
↔ 0.00158	0.00158	0.00074	0.00013	[ ]
e2d GENROU 3	0.00213	0.00012	0.00012	[ ]
↔ 0.00027	0.00027	0.00018	0.00073	[ ]
e2d GENROU 4	0.00371	0.00209	0.00209	[ ]
↔ 0.00056	0.00056	6e-05	0.0004	[ ]
e2q GENROU 1	0.00134	0.00229	0.00229	[ ]
↔ 0.00017	0.00017	0.01819	0.00503	[ ]
e2q GENROU 2	0.00325	0.01381	0.01381	[ ]
↔ 0.00019	0.00019	0.02501	0.00411	[ ]
e2q GENROU 3	0.01383	0.00437	0.00437	[ ]
↔ 0.00129	0.00129	0.00604	0.02453	[ ]
e2q GENROU 4	0.00276	0.00076	0.00076	[ ]
↔ 0.00109	0.00109	0.00287	0.01969	[ ]
LAG_y TGOV1 1	0.00067	0.0032	0.0032	[ ]
↔ 0.00728	0.00728	0.00125	0.00038	[ ]
LAG_y TGOV1 2	0.00142	0.00386	0.00386	[ ]
↔ 0.00338	0.00338	0.00175	0.00023	[ ]
LAG_y TGOV1 3	0.00355	0.00079	0.00079	[ ]
↔ 0.00371	0.00371	0.00041	0.00152	[ ]
LAG_y TGOV1 4	0.00239	0.00129	0.00129	[ ]
↔ 0.0054	0.0054	0.00019	0.00129	[ ]
LL_x TGOV1 1	3e-05	0.00016	0.00016	[ ]
↔ 0.00064	0.00064	6e-05	2e-05	[ ]
LL_x TGOV1 2	7e-05	0.0002	0.0002	[ ]
↔ 0.00031	0.00031	8e-05	1e-05	[ ]
LL_x TGOV1 3	0.00014	3e-05	3e-05	[ ]
↔ 0.00026	0.00026	1e-05	5e-05	[ ]
LL_x TGOV1 4	9e-05	5e-05	5e-05	[ ]
↔ 0.00038	0.00038	1e-05	5e-05	[ ]
vp EXDC2 1	6e-05	0.00162	0.00162	[ ]
↔ 0.00482	0.00482	0.00739	0.002	[ ]
vp EXDC2 2	0.00018	0.00133	0.00133	[ ]
↔ 0.00482	0.00482	0.01079	0.00183	[ ]
vp EXDC2 3	0.00069	0.00024	0.00024	[ ]
↔ 0.00056	0.00056	0.00174	0.00719	[ ]
vp EXDC2 4	0.00083	0.00076	0.00076	[ ]
↔ 0.00082	0.00082	0.00101	0.00706	[ ]

(continues on next page)

(continued from previous page)

LS_y EXDC2 1	1e-05	0.00022	0.00022	
↔ 0.00047	0.00047	0.00037	0.0001	
LS_y EXDC2 2	3e-05	0.00018	0.00018	
↔ 0.00046	0.00046	0.00052	9e-05	
LS_y EXDC2 3	0.00021	7e-05	7e-05	
↔ 0.00012	0.00012	0.00018	0.00074	
LS_y EXDC2 4	0.00012	0.00011	0.00011	
↔ 8e-05	8e-05	5e-05	0.00035	
LL_x EXDC2 1	0	0	0	
↔ 0	0	0	0	
LL_x EXDC2 2	0	0	0	
↔ 0	0	0	0	
LL_x EXDC2 3	0	0	0	
↔ 0	0	0	0	
LL_x EXDC2 4	0	0	0	
↔ 0	0	0	0	
LA_y EXDC2 1	1e-05	0.00021	0.00021	
↔ 0.00039	0.00039	0.00064	0.00017	
LA_y EXDC2 2	3e-05	0.00023	0.00023	
↔ 0.00052	0.00052	0.00126	0.00021	
LA_y EXDC2 3	0.00018	6e-05	6e-05	
↔ 9e-05	9e-05	0.00031	0.00125	
LA_y EXDC2 4	0.00018	0.00016	0.00016	
↔ 0.00011	0.00011	0.00015	0.00102	
W_x EXDC2 1	0	5e-05	5e-05	
↔ 0.00038	0.00038	0.00056	0.00015	
W_x EXDC2 2	1e-05	5e-05	5e-05	
↔ 0.00044	0.00044	0.00093	0.00016	
W_x EXDC2 3	3e-05	1e-05	1e-05	
↔ 7e-05	7e-05	0.00022	0.00094	
W_x EXDC2 4	3e-05	3e-05	3e-05	
↔ 9e-05	9e-05	0.00011	0.00076	

## PARTICIPATION FACTORS [4/8]

	#29	#30	#31	
↔ #32	#33	#34	#35	
delta GENROU 1	0.00184	0.00076	0.0105	
↔ 0.0105	0.00211	0.00572	0.00175	
delta GENROU 2	0.00619	0.00021	8e-05	
↔ 8e-05	0.0019	0.00509	0.00601	
delta GENROU 3	0.00524	0.00086	0.00348	
↔ 0.00348	0.00522	0.00212	0.00608	
delta GENROU 4	0.00254	0	0.00062	
↔ 0.00062	0.00562	0.00191	0.00119	
omega GENROU 1	0.00097	0.00039	0.00811	
↔ 0.00811	0.00097	0.00262	0.00049	
omega GENROU 2	0.00326	0.00011	0.00024	
↔ 0.00024	0.00087	0.00234	0.00176	
omega GENROU 3	0.00188	0.0003	0.00176	
↔ 0.00176	0.00163	0.00066	0.00119	
omega GENROU 4	0.00086	0	0.00023	

(continues on next page)

(continued from previous page)

↪ 0.00023	0.00164	0.00056	0.0002
e1q GENROU 1	0.00024	0.00202	0.07076
↪ 0.07076	0.00382	0.01402	0.05544
e1q GENROU 2	0.0016	0.0052	0.0874
↪ 0.0874	0.00481	0.01018	0.04937
e1q GENROU 3	0.00126	0.00534	0.07194
↪ 0.07194	0.01035	0.00282	0.04397
e1q GENROU 4	0.00048	0.00212	0.0524
↪ 0.0524	0.01026	0.00535	0.04396
e1d GENROU 1	0.22346	0.14687	0.02655
↪ 0.02655	0.0065	0.02563	0.01032
e1d GENROU 2	0.16265	0.15839	0.02676
↪ 0.02676	0.00999	0.01879	0.00058
e1d GENROU 3	0.1893	0.12278	0.02515
↪ 0.02515	0.02218	0.00485	0.00021
e1d GENROU 4	0.28887	0.12266	0.02446
↪ 0.02446	0.02447	0.0144	0.00922
e2d GENROU 1	1e-05	0.00011	0.00296
↪ 0.00296	0.00013	0.0005	0.00144
e2d GENROU 2	9e-05	0.00027	0.00367
↪ 0.00367	0.00016	0.00032	0.00127
e2d GENROU 3	8e-05	0.0003	0.00331
↪ 0.00331	0.00037	0.0001	0.00124
e2d GENROU 4	4e-05	0.00013	0.0025
↪ 0.0025	0.00042	0.00022	0.00131
e2q GENROU 1	0.02215	0.0174	0.00812
↪ 0.00812	0.00095	0.00378	0.00189
e2q GENROU 2	0.01605	0.01868	0.00815
↪ 0.00815	0.00146	0.00276	0.00011
e2q GENROU 3	0.01847	0.01432	0.00757
↪ 0.00757	0.00321	0.0007	4e-05
e2q GENROU 4	0.0281	0.01426	0.00734
↪ 0.00734	0.00353	0.00208	0.00165
LAG_y TGOV1 1	0.0003	0.00019	0.0023
↪ 0.0023	0.00118	0.00327	0.00534
LAG_y TGOV1 2	0.00103	5e-05	7e-05
↪ 7e-05	0.00107	0.00292	0.0193
LAG_y TGOV1 3	0.00077	0.0002	0.00065
↪ 0.00065	0.0026	0.00108	0.01706
LAG_y TGOV1 4	0.00037	0	9e-05
↪ 9e-05	0.00279	0.00096	0.00309
LL_x TGOV1 1	1e-05	1e-05	0.00065
↪ 0.00065	5e-05	0.00015	0.00014
LL_x TGOV1 2	5e-05	0	2e-05
↪ 2e-05	5e-05	0.00014	0.00054
LL_x TGOV1 3	3e-05	1e-05	0.00015
↪ 0.00015	0.0001	4e-05	0.00037
LL_x TGOV1 4	1e-05	0	2e-05
↪ 2e-05	0.0001	4e-05	7e-05
vp EXDC2 1	0.0018	0.02792	0.0755
↪ 0.0755	0.08166	0.28549	0.19372
vp EXDC2 2	0.01151	0.07037	0.09369

(continues on next page)

(continued from previous page)

↔ 0.09369	0.10072	0.20041	0.17109	
vp EXDC2 3	0.00659	0.05154	0.05539	↔
↔ 0.05539	0.15513	0.03978	0.10881	
vp EXDC2 4	0.00328	0.02678	0.05123	↔
↔ 0.05123	0.20162	0.10006	0.1402	
LS_y EXDC2 1	4e-05	0.00026	0.00452	↔
↔ 0.00452	0.00039	0.00144	0.00445	
LS_y EXDC2 2	0.00022	0.00065	0.00551	↔
↔ 0.00551	0.00047	0.00099	0.00386	
LS_y EXDC2 3	0.00027	0.00103	0.00707	↔
↔ 0.00707	0.00158	0.00043	0.00532	
LS_y EXDC2 4	7e-05	0.00026	0.00313	↔
↔ 0.00313	0.00099	0.00052	0.00329	
LL_x EXDC2 1	0	0	0	↔
↔ 0	0	0	0	
LL_x EXDC2 2	0	0	0	↔
↔ 0	0	0	0	
LL_x EXDC2 3	0	0	0	↔
↔ 0	0	0	0	
LL_x EXDC2 4	0	0	0	↔
↔ 0	0	0	0	
LA_y EXDC2 1	0.0001	0.00137	0.0025	↔
↔ 0.0025	0.00299	0.0104	0.00439	
LA_y EXDC2 2	0.0009	0.00465	0.00418	↔
↔ 0.00418	0.00497	0.00984	0.00523	
LA_y EXDC2 3	0.00078	0.00513	0.00372	↔
↔ 0.00372	0.01154	0.00294	0.00501	
LA_y EXDC2 4	0.00032	0.00221	0.00285	↔
↔ 0.00285	0.01245	0.00614	0.00536	
W_x EXDC2 1	0.00026	0.00528	0.03995	
↔ 0.03995	0.02435	0.08588	0.11819	↔
W_x EXDC2 2	0.00188	0.01518	0.05653	
↔ 0.05653	0.03424	0.06873	0.119	↔
W_x EXDC2 3	0.00159	0.01641	0.04933	
↔ 0.04933	0.07785	0.02014	0.11172	↔
W_x EXDC2 4	0.00065	0.00702	0.03754	
↔ 0.03754	0.08326	0.04168	0.11845	↔
PARTICIPATION FACTORS [5/8]				
	#36	#37	#38	
↔ #39	#40	#41	#42	
delta GENROU 1	0.00239	0.01045	0.00051	
↔ 0.00201	0.00333	0.00065	0.00065	↔
delta GENROU 2	0.00236	0.00093	0.00922	
↔ 0.00128	0.00055	0.00047	0.00047	↔
delta GENROU 3	0.00232	0.00157	0.00622	
↔ 0.00373	0.00374	0.00048	0.00048	
delta GENROU 4	0.00245	0.0139	0.00281	
↔ 0.00283	8e-05	0.00065	0.00065	↔
omega GENROU 1	0.05646	0.00087	2e-05	
↔ 7e-05	0.00453	0.00107	0.00107	↔

(continues on next page)

(continued from previous page)

omega GENROU 2	0.05461	8e-05	0.00045	[ ]
↳ 4e-05	0.00192	0.00065	0.00065	[ ]
omega GENROU 3	0.03522	8e-05	0.0002	[ ]
↳ 8e-05	0.00528	0.0005	0.0005	[ ]
omega GENROU 4	0.03404	0.0007	8e-05	[ ]
↳ 6e-05	0.00043	0.00073	0.00073	[ ]
e1q GENROU 1	0.00021	0.01108	0.00476	[ ]
↳ 0.00142	0.03388	0.08211	0.08211	[ ]
e1q GENROU 2	0.00026	0.00532	0.00231	[ ]
↳ 0.00021	0.04408	0.06594	0.06594	[ ]
e1q GENROU 3	0.0001	0.00154	0.00122	[ ]
↳ 0.0004	0.03867	0.06805	0.06805	[ ]
e1q GENROU 4	7e-05	0.00458	0.00375	[ ]
↳ 0.0018	0.02724	0.07576	0.07576	[ ]
e1d GENROU 1	0.00019	0.00414	0.00056	[ ]
↳ 0.00018	0.02258	0.00026	0.00026	[ ]
e1d GENROU 2	0.0002	0.00398	0.0024	[ ]
↳ 0.00016	0.02473	0.00126	0.00126	[ ]
e1d GENROU 3	0.0001	0.00324	0.00173	[ ]
↳ 0.00029	0.02321	0.00142	0.00142	[ ]
e1d GENROU 4	0.0001	0.0042	0.00103	[ ]
↳ 0.00044	0.02112	0.00043	0.00043	[ ]
e2d GENROU 1	0	0.00015	0.00011	[ ]
↳ 2e-05	0.00025	0.00119	0.00119	[ ]
e2d GENROU 2	0	6e-05	2e-05	[ ]
↳ 2e-05	0.00035	0.00092	0.00092	[ ]
e2d GENROU 3	0	2e-05	1e-05	[ ]
↳ 6e-05	0.00034	0.00105	0.00105	[ ]
e2d GENROU 4	0	2e-05	7e-05	[ ]
↳ 2e-05	0.00024	0.00127	0.00127	[ ]
e2q GENROU 1	4e-05	0.00084	0.00011	[ ]
↳ 4e-05	0.00566	8e-05	8e-05	[ ]
e2q GENROU 2	4e-05	0.00081	0.00049	[ ]
↳ 3e-05	0.00617	0.00037	0.00037	[ ]
e2q GENROU 3	2e-05	0.00065	0.00034	[ ]
↳ 6e-05	0.00573	0.00041	0.00041	[ ]
e2q GENROU 4	2e-05	0.00084	0.00021	[ ]
↳ 9e-05	0.0052	0.00012	0.00012	[ ]
LAG_y TGOV1 1	0.28771	0.44443	0.03358	[ ]
↳ 0.18229	0.00614	0.0005	0.0005	[ ]
LAG_y TGOV1 2	0.27828	0.03862	0.60969	[ ]
↳ 0.11639	0.0026	0.0003	0.0003	[ ]
LAG_y TGOV1 3	0.23415	0.05545	0.34788	[ ]
↳ 0.28645	0.00933	0.0003	0.0003	[ ]
LAG_y TGOV1 4	0.24068	0.49682	0.15655	[ ]
↳ 0.21672	0.0008	0.00047	0.00047	[ ]
LL_x TGOV1 1	0.02949	0.00262	0.00012	[ ]
↳ 0.00046	0.00245	0.00046	0.00046	[ ]
LL_x TGOV1 2	0.02951	0.00024	0.00224	[ ]
↳ 0.0003	0.00107	0.00029	0.00029	[ ]
LL_x TGOV1 3	0.01928	0.00026	0.00099	[ ]
↳ 0.00058	0.00299	0.00022	0.00022	[ ]

(continues on next page)

(continued from previous page)

LL_x TGOV1 4	0.0199	0.00236	0.00045	[ ]
↔ 0.00044	0.00026	0.00035	0.00035	[ ]
vp EXDC2 1	0.00028	0.01851	0.01021	[ ]
↔ 0.00225	0.02054	0.04771	0.04771	[ ]
vp EXDC2 2	0.00034	0.00805	0.00341	[ ]
↔ 0.00124	0.02771	0.03765	0.03765	[ ]
vp EXDC2 3	0.00019	0.0005	0.00126	[ ]
↔ 0.00202	0.01745	0.02788	0.02788	[ ]
vp EXDC2 4	0.00016	0.00385	0.00608	[ ]
↔ 0.00268	0.01504	0.0403	0.0403	[ ]
LS_y EXDC2 1	2e-05	0.00065	0.00035	[ ]
↔ 8e-05	0.00189	0.00251	0.00251	[ ]
LS_y EXDC2 2	2e-05	0.00028	0.00012	[ ]
↔ 4e-05	0.00251	0.00194	0.00194	[ ]
LS_y EXDC2 3	2e-05	4e-05	9e-05	[ ]
↔ 0.00015	0.00343	0.00312	0.00312	[ ]
LS_y EXDC2 4	1e-05	0.00014	0.00021	[ ]
↔ 9e-05	0.00142	0.00216	0.00216	[ ]
LL_x EXDC2 1	0	0	0	[ ]
↔ 0	0	0	0	[ ]
LL_x EXDC2 2	0	0	0	[ ]
↔ 0	0	0	0	[ ]
LL_x EXDC2 3	0	0	0	[ ]
↔ 0	0	0	0	[ ]
LL_x EXDC2 4	0	0	0	[ ]
↔ 0	0	0	0	[ ]
LA_y EXDC2 1	0	0.00028	0.00016	[ ]
↔ 4e-05	1e-05	0.00088	0.00088	[ ]
LA_y EXDC2 2	0	0.00016	7e-05	[ ]
↔ 3e-05	2e-05	0.00094	0.00094	[ ]
LA_y EXDC2 3	0	2e-05	4e-05	[ ]
↔ 6e-05	2e-05	0.00105	0.00105	[ ]
LA_y EXDC2 4	0	0.0001	0.00016	[ ]
↔ 7e-05	1e-05	0.00125	0.00125	[ ]
W_x EXDC2 1	0.0007	0.01967	0.01051	[ ]
↔ 0.00229	0.2114	0.11396	0.11396	[ ]
W_x EXDC2 2	0.00099	0.00975	0.004	[ ]
↔ 0.00144	0.32512	0.10252	0.10252	[ ]
W_x EXDC2 3	0.0008	0.00089	0.00219	[ ]
↔ 0.00345	0.30227	0.11206	0.11206	[ ]
W_x EXDC2 4	0.00057	0.00566	0.00867	[ ]
↔ 0.00377	0.21431	0.13329	0.13329	[ ]
PARTICIPATION FACTORS [6/8]				
	#43*	#44	#45	[ ]
↔ #46	#47	#48	#49	[ ]
delta GENROU 1	0.23884	0.00446	0.00446	[ ]
↔ 5e-05	5e-05	4e-05	4e-05	[ ]
delta GENROU 2	0.24952	0.00469	0.00469	[ ]
↔ 2e-05	2e-05	3e-05	3e-05	[ ]
delta GENROU 3	0.26107	0.00465	0.00465	[ ]

(continues on next page)

(continued from previous page)

→	2e-05	2e-05	7e-05	7e-05
delta GENROU 4		0.25057	0.00449	0.00449
→	7e-05	7e-05	5e-05	5e-05
omega GENROU 1		0	0.10227	0.10227
→	0.00063	0.00063	0.00032	0.00032
omega GENROU 2		0	0.10619	0.10619
→	0.00022	0.00022	0.00028	0.00028
omega GENROU 3		0	0.07177	0.07177
→	0.00013	0.00013	0.00029	0.00029
omega GENROU 4		0	0.06484	0.06484
→	0.00031	0.00031	0.0002	0.0002
e1q GENROU 1		0	2e-05	2e-05
→	0.07049	0.07049	0.04381	0.04381
e1q GENROU 2		0	2e-05	2e-05
→	0.05619	0.05619	0.05594	0.05594
e1q GENROU 3		0	3e-05	3e-05
→	0.0392	0.0392	0.06379	0.06379
e1q GENROU 4		0	4e-05	4e-05
→	0.04984	0.04984	0.04973	0.04973
e1d GENROU 1		0	0	0
→	0.00451	0.00451	0.00282	0.00282
e1d GENROU 2		0	0	0
→	0.00354	0.00354	0.00354	0.00354
e1d GENROU 3		0	0	0
→	0.00267	0.00267	0.00435	0.00435
e1d GENROU 4		0	1e-05	1e-05
→	0.00388	0.00388	0.0039	0.0039
e2d GENROU 1		0	0	0
→	0.00069	0.00069	0.00042	0.00042
e2d GENROU 2		0	0	0
→	0.00051	0.00051	0.0005	0.0005
e2d GENROU 3		0	0	0
→	0.00039	0.00039	0.00063	0.00063
e2d GENROU 4		0	0	0
→	0.00056	0.00056	0.00055	0.00055
e2q GENROU 1		0	0	0
→	0.00136	0.00136	0.00085	0.00085
e2q GENROU 2		0	0	0
→	0.00106	0.00106	0.00106	0.00106
e2q GENROU 3		0	0	0
→	0.00079	0.00079	0.00129	0.00129
e2q GENROU 4		0	0	0
→	0.00115	0.00115	0.00116	0.00116
LAG_y TGOV1 1		0	0.04715	0.04715
→	0.00028	0.00028	0.00015	0.00015
LAG_y TGOV1 2		0	0.04895	0.04895
→	0.0001	0.0001	0.00013	0.00013
LAG_y TGOV1 3		0	0.04317	0.04317
→	7e-05	7e-05	0.00017	0.00017
LAG_y TGOV1 4		0	0.04147	0.04147
→	0.0002	0.0002	0.00012	0.00012
LL_x TGOV1 1		0	0.12629	0.12629

(continues on next page)

(continued from previous page)

↔ 0.00092	0.00092	0.00049	0.00049	
LL_x TGOV1 2	0	0.13567	0.13567	↔ [ ]
↔ 0.00033	0.00033	0.00045	0.00045	
LL_x TGOV1 3	0	0.0929	0.0929	↔ [ ]
↔ 0.00019	0.00019	0.00047	0.00047	
LL_x TGOV1 4	0	0.08958	0.08958	↔ [ ]
↔ 0.00052	0.00052	0.00034	0.00034	
vp EXDC2 1	0	8e-05	8e-05	↔ [ ]
↔ 0.02856	0.02856	0.01757	0.01757	
vp EXDC2 2	0	7e-05	7e-05	↔ [ ]
↔ 0.02194	0.02194	0.02157	0.02157	
vp EXDC2 3	0	6e-05	6e-05	↔ [ ]
↔ 0.01098	0.01098	0.01765	0.01765	
vp EXDC2 4	0	8e-05	8e-05	↔ [ ]
↔ 0.01848	0.01848	0.01825	0.01825	
LS_y EXDC2 1	0	0	0	0 ↔ [ ]
↔ 0.00105	0.00105	0.00064	0.00064	
LS_y EXDC2 2	0	0	0	0 ↔ [ ]
↔ 0.00079	0.00079	0.00077	0.00077	
LS_y EXDC2 3	0	0	0	0 ↔ [ ]
↔ 0.00086	0.00086	0.00136	0.00136	
LS_y EXDC2 4	0	0	0	0 ↔ [ ]
↔ 0.00069	0.00069	0.00068	0.00068	
LL_x EXDC2 1	0	0	0	0 ↔ [ ]
↔ 0	0	0	0	
LL_x EXDC2 2	0	0	0	0 ↔ [ ]
↔ 0	0	0	0	
LL_x EXDC2 3	0	0	0	0 ↔ [ ]
↔ 0	0	0	0	
LL_x EXDC2 4	0	0	0	0 ↔ [ ]
↔ 0	0	0	0	
LA_y EXDC2 1	0	0	0	0 ↔ [ ]
↔ 0.00049	0.00049	0.0003	0.0003	
LA_y EXDC2 2	0	0	0	0 ↔ [ ]
↔ 0.00051	0.00051	0.0005	0.0005	
LA_y EXDC2 3	0	0	0	0 ↔ [ ]
↔ 0.00038	0.00038	0.00061	0.00061	
LA_y EXDC2 4	0	0	0	0 ↔ [ ]
↔ 0.00054	0.00054	0.00053	0.00053	
W_x EXDC2 1	0	0.00025	0.00025	↔ [ ]
↔ 0.11968	0.11968	0.07515	0.07515	
W_x EXDC2 2	0	0.00028	0.00028	↔ [ ]
↔ 0.10481	0.10481	0.10518	0.10518	
W_x EXDC2 3	0	0.00032	0.00032	↔ [ ]
↔ 0.07742	0.07742	0.12701	0.12701	
W_x EXDC2 4	0	0.00035	0.00035	↔ [ ]
↔ 0.10724	0.10724	0.1081	0.1081	
PARTICIPATION FACTORS [7/8]				
	#50	#51	#52	
delta GENROU 1	0.00304	0.00244	4e-05	

(continues on next page)

(continued from previous page)

delta GENROU 2	0.00157	0.00344	8e-05
delta GENROU 3	0.00179	6e-05	0.00325
delta GENROU 4	0.00266	0	0.00268
omega GENROU 1	0	0	0
omega GENROU 2	0	0	0
omega GENROU 3	0	0	0
omega GENROU 4	0	0	0
e1q GENROU 1	0	3e-05	0
e1q GENROU 2	2e-05	7e-05	0
e1q GENROU 3	0	0	6e-05
e1q GENROU 4	0	0	3e-05
e1d GENROU 1	1e-05	2e-05	0
e1d GENROU 2	0	5e-05	0
e1d GENROU 3	1e-05	0	4e-05
e1d GENROU 4	1e-05	0	3e-05
e2d GENROU 1	0	0	0
e2d GENROU 2	0	0	0
e2d GENROU 3	0	0	0
e2d GENROU 4	0	0	0
e2q GENROU 1	0	1e-05	0
e2q GENROU 2	0	1e-05	0
e2q GENROU 3	0	0	1e-05
e2q GENROU 4	0	0	1e-05
LAG_y TGOV1 1	0.00021	0.00017	0
LAG_y TGOV1 2	0.00011	0.00023	1e-05
LAG_y TGOV1 3	0.0001	0	0.00019
LAG_y TGOV1 4	0.00015	0	0.00015
LL_x TGOV1 1	0.30765	0.38484	0.0055
LL_x TGOV1 2	0.16372	0.56187	0.01273
LL_x TGOV1 3	0.12342	0.00631	0.34609
LL_x TGOV1 4	0.1836	0.00014	0.28568
vp EXDC2 1	3e-05	8e-05	0
vp EXDC2 2	2e-05	6e-05	0
vp EXDC2 3	1e-05	0	4e-05
vp EXDC2 4	2e-05	0	6e-05
LS_y EXDC2 1	0	0	0
LS_y EXDC2 2	0	0	0
LS_y EXDC2 3	0	0	0
LS_y EXDC2 4	0	0	0
LL_x EXDC2 1	0	0	0
LL_x EXDC2 2	0	0	0
LL_x EXDC2 3	0	0	0
LL_x EXDC2 4	0	0	0
LA_y EXDC2 1	0	0	0
LA_y EXDC2 2	0	0	0
LA_y EXDC2 3	0	0	0
LA_y EXDC2 4	0	0	0
W_x EXDC2 1	9e-05	0.00026	0
W_x EXDC2 2	9e-05	0.00023	1e-05
W_x EXDC2 3	5e-05	0	0.00021
W_x EXDC2 4	7e-05	0	0.00029

## 2.4.4 Parameter Sweep and Root Loci Plot

Parameter sweep allows automatically applying a set of parameters to compute the eigenvalues. It is useful to plot the root loci of the system.

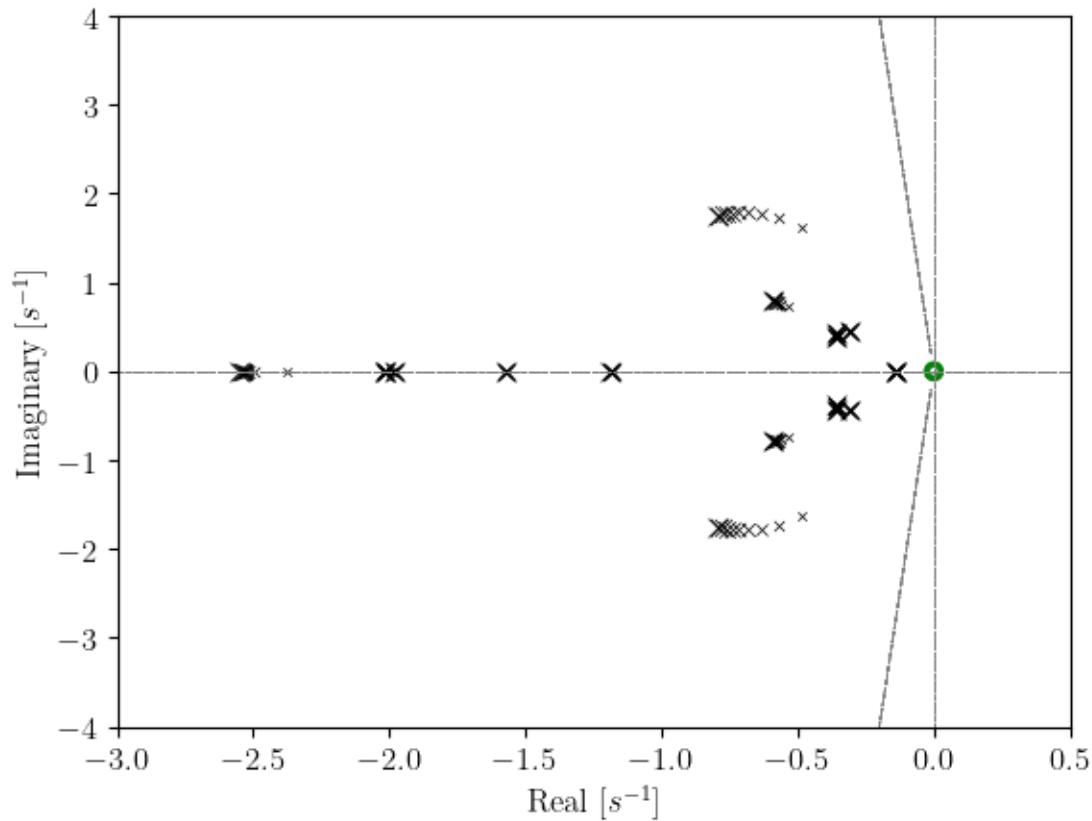
In this example, we will study the impact of EXDC2.KA of the device 1 on the system stability. Let its KA vary evenly between 20 to 200 in 10 steps.

```
ret = ss.EIG.sweep(ss.EXDC2.KA, 1, np.linspace(20, 200, 10))
```

Plot the 30th to the 50-th eigenvalues (0-based index) on the s-plane:

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

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



Note that the marker size increases linearly as the parameters sweep from the first to the last. We recommend sweeping the parameter in an ascending order so that the larger marker size correspond to a larger parameter.

## 2.4.5 Cleanup

```
!andes misc -C
```

```
"/home/hacui/repos/andes/examples/kundur_full_out.txt" removed.  
"/home/hacui/repos/andes/examples/kundur_full_eig.txt" removed.
```

## 2.5 Using CLI from Notebook

This example notebook is a supplement to the ANDES tutorial. Make sure you have read the tutorial on using the CLI first.

A brief version can be found at <https://github.com/cuihantao/andes/blob/master/README.md#run-simulations>

### 2.5.1 The ! magic in iPython

This example shows how to use the ANDES CLI from Jupyter Notebook.

It is based on the iPython magic `!.` To run a command *from within* IPython or Jupyter, place a `!` immediately before the command.

Conversely, all commands demonstrated in this notebook can be used in a terminal/shell by removing the preceding `!` sign.

### 2.5.2 Set up on Windows

Windows users will need to install **MSYS2** to support most of the Linux shell commands.

To install MSYS2-base, uncomment the following line and run it:

```
# !conda install -c msys2 -n base --yes m2-base
```

### 2.5.3 Running Shell Commands

For example, to list the directory, use `!ls`. This is equivalent to executing `ls` from the terminal.

```
!ls
```

```
demonstration ex1.ipynb ex3.ipynb ex5.ipynb ex7.ipynb ex9.ipynb  
ex10.ipynb ex2.ipynb ex4.ipynb ex6.ipynb ex8.ipynb verification
```

Likewise, to run `andes`, use `!andes`. Additional arguments can be passed as usual.

```
! andes
```

```

  _\ _ _ _ |_____| | Version 1.6.4.post10.dev0+gd1a4589d
 /_ \| ' \/_ / -_-< | Python 3.9.10 on Linux, 04/19/2022 08:30:40 PM
/_\_\_\_\_\_,\_/_/___| This program comes with ABSOLUTELY NO WARRANTY.

usage: andes [-h] [-v {1,10,20,30,40}]
           {run,plot,doc,misc,prepare,prep,selftest,st,demo} ...

positional arguments:
  {run,plot,doc,misc,prepare,prep,selftest,st,demo}
    [run] run simulation routine; [plot] plot results;
    [doc] quick documentation; [misc] misc. functions;
    [prepare] prepare the numerical code; [selftest] run
    self test;

optional arguments:
  -h, --help            show this help message and exit
  -v {1,10,20,30,40}, --verbose {1,10,20,30,40}
                        Verbosity level in 10-DEBUG, 20-INFO, 30-WARNING, or
                        40-ERROR.
```

## 2.5.4 Run a simulation

Pass the path to the case file and other arguments to andes from the command line as follows.

```
! andes run ../andes/cases/kundur/kundur_full.xlsx -r tds
```

```

  _\ _ _ _ |_____| | Version 1.6.4.post10.dev0+gd1a4589d
 /_ \| ' \/_ / -_-< | Python 3.9.10 on Linux, 04/19/2022 08:30:41 PM
/_\_\_\_\_\_,\_/_/___| This program comes with ABSOLUTELY NO WARRANTY.

Working directory: "/home/hacui/repos/andes/examples"
> Loaded config from file "/home/hacui/.andes/andes.rc"
> Loaded generated Python code in "/home/hacui/.andes/pycode".
Parsing input file "../andes/cases/kundur/kundur_full.xlsx"...
Input file parsed in 0.2502 seconds.
System internal structure set up in 0.0306 seconds.
-> System connectivity check results:
  No islanded bus detected.
  System is interconnected.
  Each island has a slack bus correctly defined and enabled.

-> Power flow calculation
  Numba: Off
  Sparse solver: KLU
  Solution method: NR method
Power flow initialized in 0.0029 seconds.
```

(continues on next page)

(continued from previous page)

```

0: |F(x)| = 14.9282832
1: |F(x)| = 3.608627841
2: |F(x)| = 0.1701107882
3: |F(x)| = 0.002038626956
4: |F(x)| = 3.745104027e-07
Converged in 5 iterations in 0.0032 seconds.
Initialization for dynamics completed in 0.0278 seconds.
Initialization was successful.
Report saved to "kundur_full_out.txt" in 0.0016 seconds.

-> Time Domain Simulation Summary:
Sparse Solver: KLU
Simulation time: 0.0-20.0 s.
Fixed step size: h=33.33 ms. Shrink if not converged.
<Toggle 1>: Line.Line_8 status changed to 0 at t=2.0 sec.
100%|#####
Simulation completed in 1.1546 seconds.
Outputs to "kundur_full_out.lst" and "kundur_full_out.npz".
Outputs written in 0.0235 seconds.
-> Single process finished in 1.6411 seconds.

```

Case file names can be separated from the path, which can be passed to `-p`. The above command is equivalent to

```
!andes run kundur_full.xlsx -p "../andes/cases/kundur/" -r tds
```

```

/ \ _ _ _ _ | | _ _ _ | Version 1.6.4.post10.dev0+gd1a4589d
/ _ \| ' \ / _` / -_-|-< | Python 3.9.10 on Linux, 04/19/2022 08:30:43 PM
/ / \ \_||\_,_ \_/_/ | This program comes with ABSOLUTELY NO WARRANTY.

Working directory: "/home/hacui/repos/andes/examples"
> Loaded config from file "/home/hacui/.andes/andes.rc"
> Loaded generated Python code in "/home/hacui/.andes/pycode".
Parsing input file "../andes/cases/kundur/kundur_full.xlsx"...
Input file parsed in 0.2501 seconds.
System internal structure set up in 0.0305 seconds.
-> System connectivity check results:
  No islanded bus detected.
  System is interconnected.
  Each island has a slack bus correctly defined and enabled.

-> 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

```

(continues on next page)

(continued from previous page)

```

3: |F(x)| = 0.002038626956
4: |F(x)| = 3.745104027e-07
Converged in 5 iterations in 0.0032 seconds.
Initialization for dynamics completed in 0.0277 seconds.
Initialization was successful.
Report saved to "kundur_full_out.txt" in 0.0016 seconds.

-> Time Domain Simulation Summary:
Sparse Solver: KLU
Simulation time: 0.0-20.0 s.
Fixed step size: h=33.33 ms. Shrink if not converged.
<Toggle 1>: Line.Line_8 status changed to 0 at t=2.0 sec.
100%|#####
Simulation completed in 1.1584 seconds.
Outputs to "kundur_full_out.lst" and "kundur_full_out.npz".
Outputs written in 0.0237 seconds.
-> Single process finished in 1.6446 seconds.

```

```
!pwd
```

```
/home/hacui/repos/andes/examples
```

```

import os

os.path.isfile('../andes/cases/kundur/kundur_full.xlsx')

```

```
True
```

## PSS/E RAW and DYR Files

To run a simulation using PSS/E raw and dyr files, pass the dyr file to argument `--addfile`.

For example:

```
!andes run ../andes/cases/kundur/kundur.raw --addfile ../andes/cases/kundur/
→kundur_full.dyr -r tds
```

```

          | Version 1.6.4.post10.dev0+gd1a4589d
  / \ _ _ _ | |__ __ | Python 3.9.10 on Linux, 04/19/2022 08:30:46 PM
  / _ \| ' \/ _` / -_-|-< |
  /_ \ \_\_|_|\_,_\_\_/_/ | This program comes with ABSOLUTELY NO WARRANTY.

Working directory: "/home/hacui/repos/andes/examples"
> Loaded config from file "/home/hacui/.andes/andes.rc"
> Loaded generated Python code in "/home/hacui/.andes/pycode".
Parsing input file "../andes/cases/kundur/kundur.raw"...
  MODIFIED KUNDUR'S TWO-AREA TEST SYSTEM, DISTRIBUTED WITH ANDES

```

(continues on next page)

(continued from previous page)

SEE THE BOOK "POWER SYSTEM STABILITY AND CONTROL" FOR ORIGINAL DATA  
Input file parsed in 0.0024 seconds.  
Parsing additional file ".../andes/cases/kundur/kundur\_full.dyr"...  
Addfile parsed in 0.2239 seconds.  
System internal structure set up in 0.0297 seconds.  
-> System connectivity check results:  
No islanded bus detected.  
System is interconnected.  
Each island has a slack bus correctly defined and enabled.

-> Power flow calculation  
Numba: Off  
Sparse solver: KLU  
Solution method: NR method  
Power flow initialized in 0.0028 seconds.  
0: |F(x)| = 3.175850023  
1: |F(x)| = 3.176155228e-08  
Converged in 2 iterations in 0.0014 seconds.  
Initialization for dynamics completed in 0.0276 seconds.  
Initialization was successful.  
Report saved to "kundur\_out.txt" in 0.0014 seconds.

-> Time Domain Simulation Summary:  
Sparse Solver: KLU  
Simulation time: 0.0-20.0 s.  
Fixed step size: h=33.33 ms. Shrink if not converged.  
<Toggle Toggle\_1>: Line.Line\_8 status changed to 0 at t=2.0 sec.  
100%|#####| 100.0/100 [00:01<00:00, 86.44%/s]  
Simulation completed in 1.1569 seconds.  
Outputs to "kundur\_out.lst" and "kundur\_out.npz".  
Outputs written in 0.0235 seconds.  
-> Single process finished in 1.6165 seconds.

## 2.5.5 Check the output 1st file

```
!cat kundur_full_out.lst
```

0,	Time [s],	Time [s]
1,	delta GENROU 1,	\$\delta GENROU 1
2,	delta GENROU 2,	\$\delta GENROU 2
3,	delta GENROU 3,	\$\delta GENROU 3
4,	delta GENROU 4,	\$\delta GENROU 4
5,	omega GENROU 1,	\$\omega GENROU 1
6,	omega GENROU 2,	\$\omega GENROU 2
7,	omega GENROU 3,	\$\omega GENROU 3
8,	omega GENROU 4,	\$\omega GENROU 4
9,	e1q GENROU 1,	\$e'_q GENROU 1
10,	e1q GENROU 2,	\$e'_q GENROU 2

(continues on next page)

(continued from previous page)

11,	e1q GENROU 3,	\$e'_q\$ GENROU 3
12,	e1q GENROU 4,	\$e'_q\$ GENROU 4
13,	e1d GENROU 1,	\$e'_d\$ GENROU 1
14,	e1d GENROU 2,	\$e'_d\$ GENROU 2
15,	e1d GENROU 3,	\$e'_d\$ GENROU 3
16,	e1d GENROU 4,	\$e'_d\$ GENROU 4
17,	e2d GENROU 1,	\$e''_d\$ GENROU 1
18,	e2d GENROU 2,	\$e''_d\$ GENROU 2
19,	e2d GENROU 3,	\$e''_d\$ GENROU 3
20,	e2d GENROU 4,	\$e''_d\$ GENROU 4
21,	e2q GENROU 1,	\$e''_q\$ GENROU 1
22,	e2q GENROU 2,	\$e''_q\$ GENROU 2
23,	e2q GENROU 3,	\$e''_q\$ GENROU 3
24,	e2q GENROU 4,	\$e''_q\$ GENROU 4
25,	LAG_y TGOV1 1,	\$y_{LAG}\$_\$ TGOV1 1
26,	LAG_y TGOV1 2,	\$y_{LAG}\$_\$ TGOV1 2
27,	LAG_y TGOV1 3,	\$y_{LAG}\$_\$ TGOV1 3
28,	LAG_y TGOV1 4,	\$y_{LAG}\$_\$ TGOV1 4
29,	LL_x TGOV1 1,	\$x'_{LL}\$_\$ TGOV1 1
30,	LL_x TGOV1 2,	\$x'_{LL}\$_\$ TGOV1 2
31,	LL_x TGOV1 3,	\$x'_{LL}\$_\$ TGOV1 3
32,	LL_x TGOV1 4,	\$x'_{LL}\$_\$ TGOV1 4
33,	vp EXDC2 1,	\$V_p\$ EXDC2 1
34,	vp EXDC2 2,	\$V_p\$ EXDC2 2
35,	vp EXDC2 3,	\$V_p\$ EXDC2 3
36,	vp EXDC2 4,	\$V_p\$ EXDC2 4
37,	LS_y EXDC2 1,	\$y_{LS}\$_\$ EXDC2 1
38,	LS_y EXDC2 2,	\$y_{LS}\$_\$ EXDC2 2
39,	LS_y EXDC2 3,	\$y_{LS}\$_\$ EXDC2 3
40,	LS_y EXDC2 4,	\$y_{LS}\$_\$ EXDC2 4
41,	LL_x EXDC2 1,	\$x'_{LL}\$_\$ EXDC2 1
42,	LL_x EXDC2 2,	\$x'_{LL}\$_\$ EXDC2 2
43,	LL_x EXDC2 3,	\$x'_{LL}\$_\$ EXDC2 3
44,	LL_x EXDC2 4,	\$x'_{LL}\$_\$ EXDC2 4
45,	LA_y EXDC2 1,	\$y_{LA}\$_\$ EXDC2 1
46,	LA_y EXDC2 2,	\$y_{LA}\$_\$ EXDC2 2
47,	LA_y EXDC2 3,	\$y_{LA}\$_\$ EXDC2 3
48,	LA_y EXDC2 4,	\$y_{LA}\$_\$ EXDC2 4
49,	W_x EXDC2 1,	\$x'_{W}\$_\$ EXDC2 1
50,	W_x EXDC2 2,	\$x'_{W}\$_\$ EXDC2 2
51,	W_x EXDC2 3,	\$x'_{W}\$_\$ EXDC2 3
52,	W_x EXDC2 4,	\$x'_{W}\$_\$ EXDC2 4
53,	a Bus 1,	\$\theta\$ Bus 1
54,	a Bus 2,	\$\theta\$ Bus 2
55,	a Bus 3,	\$\theta\$ Bus 3
56,	a Bus 4,	\$\theta\$ Bus 4
57,	a Bus 5,	\$\theta\$ Bus 5
58,	a Bus 6,	\$\theta\$ Bus 6
59,	a Bus 7,	\$\theta\$ Bus 7
60,	a Bus 8,	\$\theta\$ Bus 8
61,	a Bus 9,	\$\theta\$ Bus 9
62,	a Bus 10,	\$\theta\$ Bus 10

(continues on next page)

(continued from previous page)

63,	v Bus 1,	\$V\$ Bus 1
64,	v Bus 2,	\$V\$ Bus 2
65,	v Bus 3,	\$V\$ Bus 3
66,	v Bus 4,	\$V\$ Bus 4
67,	v Bus 5,	\$V\$ Bus 5
68,	v Bus 6,	\$V\$ Bus 6
69,	v Bus 7,	\$V\$ Bus 7
70,	v Bus 8,	\$V\$ Bus 8
71,	v Bus 9,	\$V\$ Bus 9
72,	v Bus 10,	\$V\$ Bus 10
73,	q PV 2,	\$q\$ PV 2
74,	q PV 3,	\$q\$ PV 3
75,	q PV 4,	\$q\$ PV 4
76,	q Slack 1,	\$q\$ Slack 1
77,	p Slack 1,	\$p\$ Slack 1
78,	Id GENROU 1,	\$I_d\$ GENROU 1
79,	Id GENROU 2,	\$I_d\$ GENROU 2
80,	Id GENROU 3,	\$I_d\$ GENROU 3
81,	Id GENROU 4,	\$I_d\$ GENROU 4
82,	Iq GENROU 1,	\$I_q\$ GENROU 1
83,	Iq GENROU 2,	\$I_q\$ GENROU 2
84,	Iq GENROU 3,	\$I_q\$ GENROU 3
85,	Iq GENROU 4,	\$I_q\$ GENROU 4
86,	vd GENROU 1,	\$V_d\$ GENROU 1
87,	vd GENROU 2,	\$V_d\$ GENROU 2
88,	vd GENROU 3,	\$V_d\$ GENROU 3
89,	vd GENROU 4,	\$V_d\$ GENROU 4
90,	vq GENROU 1,	\$V_q\$ GENROU 1
91,	vq GENROU 2,	\$V_q\$ GENROU 2
92,	vq GENROU 3,	\$V_q\$ GENROU 3
93,	vq GENROU 4,	\$V_q\$ GENROU 4
94,	tm GENROU 1,	\$\tau_m\$ GENROU 1
95,	tm GENROU 2,	\$\tau_m\$ GENROU 2
96,	tm GENROU 3,	\$\tau_m\$ GENROU 3
97,	tm GENROU 4,	\$\tau_m\$ GENROU 4
98,	te GENROU 1,	\$\tau_e\$ GENROU 1
99,	te GENROU 2,	\$\tau_e\$ GENROU 2
100,	te GENROU 3,	\$\tau_e\$ GENROU 3
101,	te GENROU 4,	\$\tau_e\$ GENROU 4
102,	vf GENROU 1,	\$v_f\$ GENROU 1
103,	vf GENROU 2,	\$v_f\$ GENROU 2
104,	vf GENROU 3,	\$v_f\$ GENROU 3
105,	vf GENROU 4,	\$v_f\$ GENROU 4
106,	XadIfd GENROU 1,	\$X_{ad}I_{fd}\$ GENROU 1
107,	XadIfd GENROU 2,	\$X_{ad}I_{fd}\$ GENROU 2
108,	XadIfd GENROU 3,	\$X_{ad}I_{fd}\$ GENROU 3
109,	XadIfd GENROU 4,	\$X_{ad}I_{fd}\$ GENROU 4
110,	Pe GENROU 1,	\$P_e\$ GENROU 1
111,	Pe GENROU 2,	\$P_e\$ GENROU 2
112,	Pe GENROU 3,	\$P_e\$ GENROU 3
113,	Pe GENROU 4,	\$P_e\$ GENROU 4
114,	Qe GENROU 1,	\$Q_e\$ GENROU 1

(continues on next page)

(continued from previous page)

115,	Qe GENROU 2,	\$Q_e\$ GENROU 2
116,	Qe GENROU 3,	\$Q_e\$ GENROU 3
117,	Qe GENROU 4,	\$Q_e\$ GENROU 4
118,	psid GENROU 1,	\$\psi_d\$ GENROU 1
119,	psid GENROU 2,	\$\psi_d\$ GENROU 2
120,	psid GENROU 3,	\$\psi_d\$ GENROU 3
121,	psid GENROU 4,	\$\psi_d\$ GENROU 4
122,	psiq GENROU 1,	\$\psi_q\$ GENROU 1
123,	psiq GENROU 2,	\$\psi_q\$ GENROU 2
124,	psiq GENROU 3,	\$\psi_q\$ GENROU 3
125,	psiq GENROU 4,	\$\psi_q\$ GENROU 4
126,	psi2q GENROU 1,	\$\psi_{2q}\$ GENROU 1
127,	psi2q GENROU 2,	\$\psi_{2q}\$ GENROU 2
128,	psi2q GENROU 3,	\$\psi_{2q}\$ GENROU 3
129,	psi2q GENROU 4,	\$\psi_{2q}\$ GENROU 4
130,	psi2d GENROU 1,	\$\psi_{2d}\$ GENROU 1
131,	psi2d GENROU 2,	\$\psi_{2d}\$ GENROU 2
132,	psi2d GENROU 3,	\$\psi_{2d}\$ GENROU 3
133,	psi2d GENROU 4,	\$\psi_{2d}\$ GENROU 4
134,	psi2 GENROU 1,	\$\psi_2\$ GENROU 1
135,	psi2 GENROU 2,	\$\psi_2\$ GENROU 2
136,	psi2 GENROU 3,	\$\psi_2\$ GENROU 3
137,	psi2 GENROU 4,	\$\psi_2\$ GENROU 4
138,	Se GENROU 1,	\$S_e(\psi_a)\$ GENROU 1
139,	Se GENROU 2,	\$S_e(\psi_a)\$ GENROU 2
140,	Se GENROU 3,	\$S_e(\psi_a)\$ GENROU 3
141,	Se GENROU 4,	\$S_e(\psi_a)\$ GENROU 4
142,	XaqI1q GENROU 1,	\$X_{aq}I_{1q}\$ GENROU 1
143,	XaqI1q GENROU 2,	\$X_{aq}I_{1q}\$ GENROU 2
144,	XaqI1q GENROU 3,	\$X_{aq}I_{1q}\$ GENROU 3
145,	XaqI1q GENROU 4,	\$X_{aq}I_{1q}\$ GENROU 4
146,	paux TGOV1 1,	\$P_{aux}\$ TGOV1 1
147,	paux TGOV1 2,	\$P_{aux}\$ TGOV1 2
148,	paux TGOV1 3,	\$P_{aux}\$ TGOV1 3
149,	paux TGOV1 4,	\$P_{aux}\$ TGOV1 4
150,	pout TGOV1 1,	\$P_{out}\$ TGOV1 1
151,	pout TGOV1 2,	\$P_{out}\$ TGOV1 2
152,	pout TGOV1 3,	\$P_{out}\$ TGOV1 3
153,	pout TGOV1 4,	\$P_{out}\$ TGOV1 4
154,	wref TGOV1 1,	\$\omega_{ref}\$ TGOV1 1
155,	wref TGOV1 2,	\$\omega_{ref}\$ TGOV1 2
156,	wref TGOV1 3,	\$\omega_{ref}\$ TGOV1 3
157,	wref TGOV1 4,	\$\omega_{ref}\$ TGOV1 4
158,	pref TGOV1 1,	\$P_{ref}\$ TGOV1 1
159,	pref TGOV1 2,	\$P_{ref}\$ TGOV1 2
160,	pref TGOV1 3,	\$P_{ref}\$ TGOV1 3
161,	pref TGOV1 4,	\$P_{ref}\$ TGOV1 4
162,	wd TGOV1 1,	\$\omega_{dev}\$ TGOV1 1
163,	wd TGOV1 2,	\$\omega_{dev}\$ TGOV1 2
164,	wd TGOV1 3,	\$\omega_{dev}\$ TGOV1 3
165,	wd TGOV1 4,	\$\omega_{dev}\$ TGOV1 4
166,	pd TGOV1 1,	\$P_d\$ TGOV1 1

(continues on next page)

(continued from previous page)

167,	pd TGOV1 2,	\$P_d\$ TGOV1 2
168,	pd TGOV1 3,	\$P_d\$ TGOV1 3
169,	pd TGOV1 4,	\$P_d\$ TGOV1 4
170,	LL_y TGOV1 1,	\$y_{\{LL\}}\$ TGOV1 1
171,	LL_y TGOV1 2,	\$y_{\{LL\}}\$ TGOV1 2
172,	LL_y TGOV1 3,	\$y_{\{LL\}}\$ TGOV1 3
173,	LL_y TGOV1 4,	\$y_{\{LL\}}\$ TGOV1 4
174,	v EXDC2 1,	\$E_{\{term\}}\$ EXDC2 1
175,	v EXDC2 2,	\$E_{\{term\}}\$ EXDC2 2
176,	v EXDC2 3,	\$E_{\{term\}}\$ EXDC2 3
177,	v EXDC2 4,	\$E_{\{term\}}\$ EXDC2 4
178,	vout EXDC2 1,	\$v_{\{out\}}\$ EXDC2 1
179,	vout EXDC2 2,	\$v_{\{out\}}\$ EXDC2 2
180,	vout EXDC2 3,	\$v_{\{out\}}\$ EXDC2 3
181,	vout EXDC2 4,	\$v_{\{out\}}\$ EXDC2 4
182,	vref EXDC2 1,	\$V_{\{ref\}}\$ EXDC2 1
183,	vref EXDC2 2,	\$V_{\{ref\}}\$ EXDC2 2
184,	vref EXDC2 3,	\$V_{\{ref\}}\$ EXDC2 3
185,	vref EXDC2 4,	\$V_{\{ref\}}\$ EXDC2 4
186,	Se EXDC2 1,	\$S_e( V_{\{out\}} )\$ EXDC2 1
187,	Se EXDC2 2,	\$S_e( V_{\{out\}} )\$ EXDC2 2
188,	Se EXDC2 3,	\$S_e( V_{\{out\}} )\$ EXDC2 3
189,	Se EXDC2 4,	\$S_e( V_{\{out\}} )\$ EXDC2 4
190,	vi EXDC2 1,	\$V_i\$ EXDC2 1
191,	vi EXDC2 2,	\$V_i\$ EXDC2 2
192,	vi EXDC2 3,	\$V_i\$ EXDC2 3
193,	vi EXDC2 4,	\$V_i\$ EXDC2 4
194,	LL_y EXDC2 1,	\$y_{\{LL\}}\$ EXDC2 1
195,	LL_y EXDC2 2,	\$y_{\{LL\}}\$ EXDC2 2
196,	LL_y EXDC2 3,	\$y_{\{LL\}}\$ EXDC2 3
197,	LL_y EXDC2 4,	\$y_{\{LL\}}\$ EXDC2 4
198,	w_y EXDC2 1,	\$y_w\$ EXDC2 1
199,	w_y EXDC2 2,	\$y_w\$ EXDC2 2
200,	w_y EXDC2 3,	\$y_w\$ EXDC2 3
201,	w_y EXDC2 4,	\$y_w\$ EXDC2 4

## 2.5.6 Plot and save to file

We found a limitation of using `andes plot` from within Notebook/iPython. The figure won't be displayed correctly. The workaround is to save the image as a file and display it from the notebook.

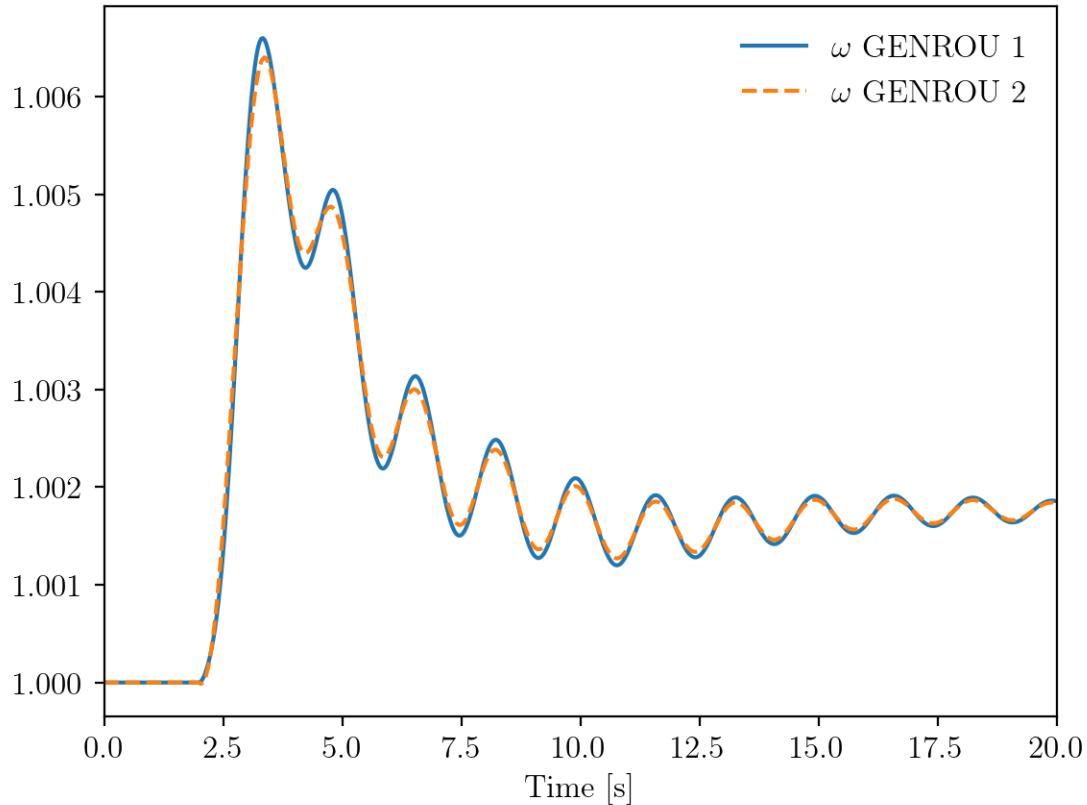
Please let us know if you have better solutions.

```
!andes plot kundur_full_out.lst 0 5 6 --save
```

```
Figure saved to "kundur_full_out_1.png".
Figure(640x480)
```

## 2.5.7 Display image

```
from IPython.display import Image
Image("kundur_full_out_1.png")
```



## 2.5.8 Using xargs for index lookup

A convenient tool in Linux/macOS is `xargs`, which turns the standard output of one program into arguments for another.

`andes plot --xargs` accepts an input of search pattern for variable names and returns a list of arguments, including the matched indices, that can be understood by `andes plot`.

To illustrate, let's look at an example output of `andes plot --xargs`.

```
!andes plot kundur_full_out.lst --xargs "omega GENROU"
```

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

The output consists of the lst file name, the default x-axis index 0, and the indices for the found variables. The full output can be passed to `andes plot` without modification.

We use the following command to pass the arguments:

```
!andes plot kundur_full_out.lst --xargs "omega GENROU" | xargs andes plot
```

Figure(640x480)

where `|` is the pipe operator in shell for piping the standard output of the left-hand side to the right-hand side, `xargs` captures the pipe-in and appends it to `andes plot`.

The command is equivalent to manually running

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

Figure(640x480)

## 2.5.9 Cleanup

Remove the saved png image files.

```
!rm -v *.png
```

```
removed 'kundur_full_out_1.png'
```

```
!andes misc -C
```

```
"/home/hacui/repos/andes/examples/kundur_full_out.npz" removed.  
"/home/hacui/repos/andes/examples/kundur_full_out.txt" removed.  
"/home/hacui/repos/andes/examples/kundur_out.lst" removed.  
"/home/hacui/repos/andes/examples/kundur_out.npz" removed.  
"/home/hacui/repos/andes/examples/kundur_full_out.lst" removed.  
"/home/hacui/repos/andes/examples/kundur_out.txt" removed.
```

## 2.6 Batch Processing - Generate Cases

This notebook demonstrates creating cases in batch and running them in parallel.

### 2.6.1 Create Cases in Batch

The approach to create cases in batch following this procedure:

- Load the base case from file
- For each desired case output:
  - Alter parameters to the desired value
  - Save each system to a new case file

```
import andes
import numpy as np
from andes.utils.paths import get_case

andes.config_logger(stream_level=30) # brief logging
```

```
# create directory for output cases
!rm -rf batch_cases
!mkdir -p batch_cases
```

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

We demonstrate running the Kundur's system under different loading conditions.

Cases are created by modifying the p0 of PQ with idx == PQ\_0.

As always, input parameters can be inspected by accessing Model.as\_df(vin=True).

```
p0_base = ss.PQ.get('p0', "PQ_0")
```

Create 3 cases so that the load increases from p0\_base to 1.2 \* p0\_base.

```
N_CASES = 3 # Note: increase `N_CASES` as necessary

p0_values = np.linspace(p0_base, 1.2 * p0_base, N_CASES)
```

```
for value in p0_values:
    ss.PQ.alter('p0', 'PQ_0', value)
    file_name = f'batch_cases/kundur_p_{value:.2f}.xlsx'

    andes.io.dump(ss, 'xlsx', file_name, overwrite=True)
```

## 2.6.2 Parallel Simulation

Parallel simulation is easy with the command line tool.

Change directory to batch\_cases:

```
import os

# change the Python working directory
os.chdir('batch_cases')
```

```
!ls -la
```

```
total 56
drwxr-xr-x 2 hacui hacui 4096 Apr 19 20:31 .
drwxr-xr-x 6 hacui hacui 4096 Apr 19 20:31 ..
```

(continues on next page)

(continued from previous page)

```
-rw-r--r-- 1 hacui hacui 14841 Apr 19 20:31 kundur_p_11.59.xlsx
-rw-r--r-- 1 hacui hacui 14842 Apr 19 20:31 kundur_p_12.75.xlsx
-rw-r--r-- 1 hacui hacui 14840 Apr 19 20:31 kundur_p_13.91.xlsx
```

## Running from Command line

```
!andes run *.xlsx -r tds
```

```
          | Version 1.6.4.post10.dev0+gd1a4589d
 /_\ _ _ _ | |_____| | Python 3.9.10 on Linux, 04/19/2022 08:31:06 PM
 /_ \| ' \/_` /_-|_-< |
 /_\_\_||\_,\_\\__/_/ | This program comes with ABSOLUTELY NO WARRANTY.

Working directory: "/home/hacui/repos/andes/examples/batch_cases"
-> Processing 3 jobs on 12 CPUs.
Process 0 for "kundur_p_11.59.xlsx" started.
Process 1 for "kundur_p_12.75.xlsx" started.
Process 2 for "kundur_p_13.91.xlsx" started.
<Toggle 1>: Line.Line_8 status changed to 0 at t=2.0 sec.
<Toggle 1>: Line.Line_8 status changed to 0 at t=2.0 sec.
<Toggle 1>: Line.Line_8 status changed to 0 at t=2.0 sec.
100%|#####| 100.0/100 [00:01<00:00, 89.55%/
 ↵s]
100%|#####| 100.0/100 [00:01<00:00, 87.15%/
 ↵s]
100%|#####| 100.0/100 [00:01<00:00, 83.73%/
 ↵s]
Log saved to "/tmp/andes-4fbaduja/andes.log".
-> Multiprocessing finished in 1.7566 seconds.
```

## Number of CPUs

In some cases, you don't want the simulation to use up all resources.

ANDES allows to control the number of processes to run in parallel through `--ncpu NCPU`, where NCPU is the maximum number of processes (equivalent to the number of CPU cores) allowed.

```
!andes run *.xlsx -r tds --ncpu 4
```

```
          | Version 1.6.4.post10.dev0+gd1a4589d
 /_\ _ _ _ | |_____| | Python 3.9.10 on Linux, 04/19/2022 08:31:09 PM
 /_ \| ' \/_` /_-|_-< |
 /_\_\_||\_,\_\\__/_/ | This program comes with ABSOLUTELY NO WARRANTY.

Working directory: "/home/hacui/repos/andes/examples/batch_cases"
-> Processing 3 jobs on 4 CPUs.
Process 0 for "kundur_p_11.59.xlsx" started.
```

(continues on next page)

(continued from previous page)

```

Process 1 for "kundur_p_12.75.xlsx" started.
Process 2 for "kundur_p_13.91.xlsx" started.
<Toggle 1>: Line.Line_8 status changed to 0 at t=2.0 sec.
<Toggle 1>: Line.Line_8 status changed to 0 at t=2.0 sec.
<Toggle 1>: Line.Line_8 status changed to 0 at t=2.0 sec.
100%|#####
100%|#####
100%|#####
Log saved to "/tmp/andes-rlpeyxo9/andes.log".
-> Multiprocessing finished in 1.7693 seconds.

```

## Running with APIs

Setting `pool = True` allows returning all system instances in a list.

This comes with a penalty in computation time but can be helpful if you want to extract data directly.

```
systems = andes.run('*xlsx', routine='tds', pool=True, verbose=10)
```

```

Working directory: "/home/hacui/repos/andes/examples/batch_cases"
Found files: ['kundur_p_11.59.xlsx', 'kundur_p_12.75.xlsx', 'kundur_p_13.91.xlsx']
-> Processing 3 jobs on 12 CPUs.

```

```

Cases are processed in the following order:
"kundur_p_11.59.xlsx"
"kundur_p_12.75.xlsx"
"kundur_p_13.91.xlsx"
<Toggle 1>: Line.Line_8 status changed to 0 at t=2.0 sec.<Toggle 1>: Line.
->Line_8 status changed to 0 at t=2.0 sec.

<Toggle 1>: Line.Line_8 status changed to 0 at t=2.0 sec.
Log saved to "/tmp/andes-lmohp4sw/andes.log".
-> Multiprocessing finished in 3.2217 seconds.

```

```
systems[0]
```

```
<andes.system.System at 0x7fbaa050a070>
```

```
systems
```

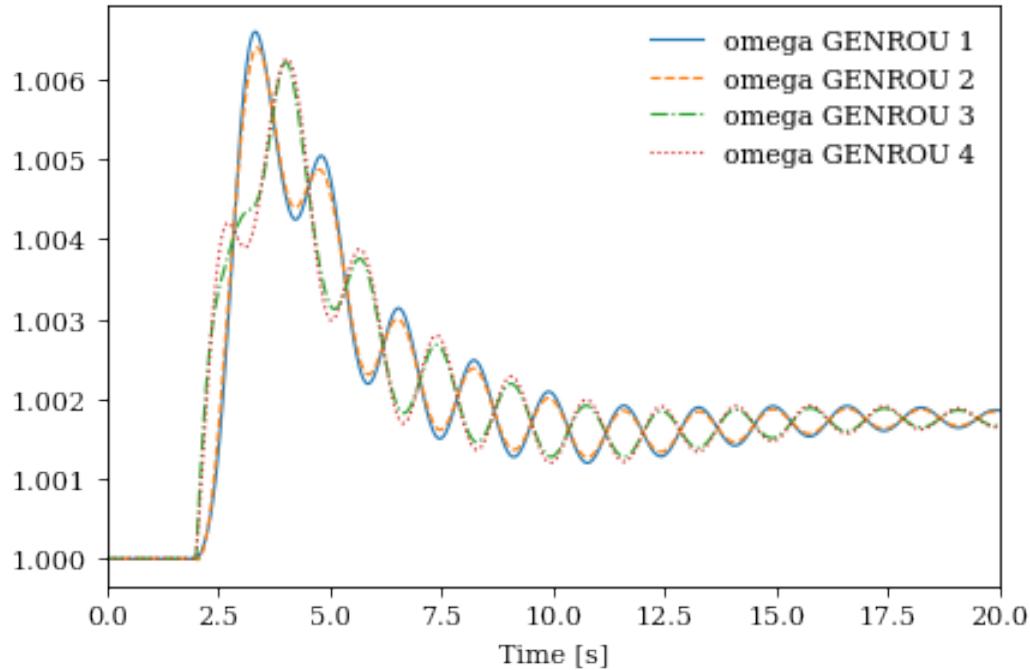
```
[<andes.system.System at 0x7fbaa050a070>,
 <andes.system.System at 0x7fba7bca2c70>,
 <andes.system.System at 0x7fba7bc9db80>]
```

## Example plots

Plotting or data analyses can be carried out as usual.

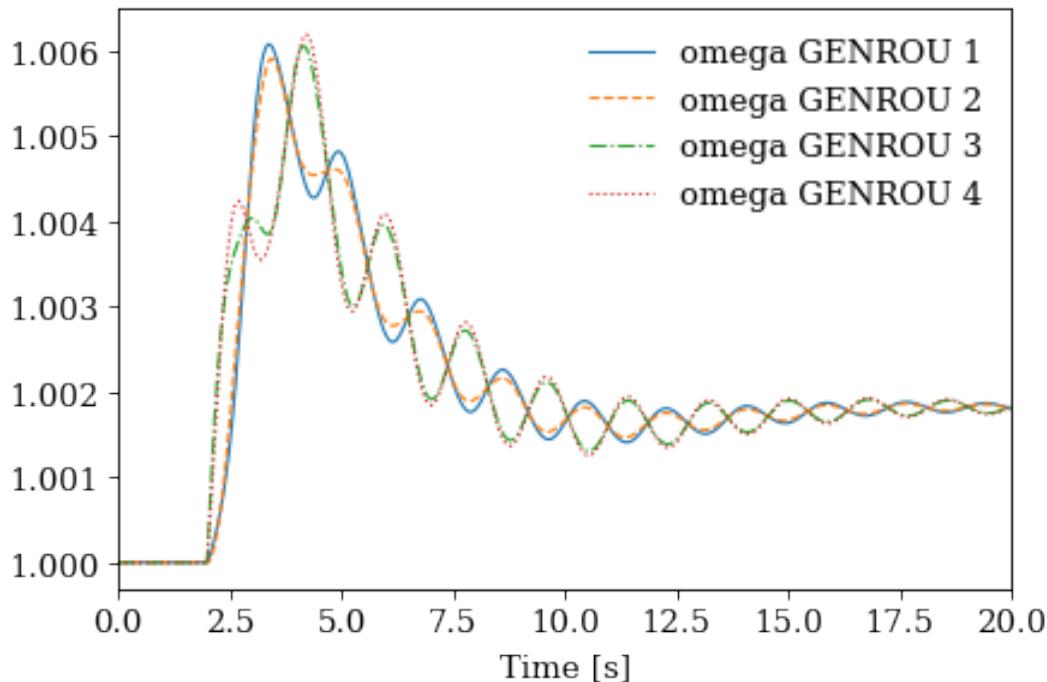
```
ss = systems[0]
```

```
systems[0].TDS.plotter.plot(ss.GENROU.omega, latex=False)
```



```
(<Figure size 480x320 with 1 Axes>, <AxesSubplot:xlabel='Time [s]'>)
```

```
systems[2].TDS.plotter.plot(ss.GENROU.omega, latex=False)
```



```
(<Figure size 480x320 with 1 Axes>, <AxesSubplot:xlabel='Time [s]'>)
```

```
!andes misc -C
!rm -rf batch_cases
```

```
No output file found in the working directory.
```

## 2.7 Batch Processing - in Memory

This notebook shows examples for batch power flow and time-domain calculations. Readers are supposed to have read the previous examples, especially Example 7 for parallel simulations.

```
import andes
import numpy as np

from matplotlib import pyplot as plt
```

## 2.7.1 Batch Power Flow Calculation

Use the Kundur's system as the example. Suppose we want to calculate power flow for the same system structure but for different load levels.

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

```
ss = andes.run(kundur,
                 no_output=True,
                 default_config=True,
                 verbose=30, # set logging level to WARNING
                 )
```

```
-> Single process finished in 0.4701 seconds.
```

There are two PQ loads in the Kundur's system with idxes of PQ\_0 and PQ\_1.

```
ss.PQ.as_df(vin=True)
```

uid	idx	u	name	bus	Vn	p0	q0	vmax	vmin	owner
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

If we have a range of active power for each load, such as

```
n_samples = 3 # Note: increase `n_samples` for higher data resolution

pq0_values = np.linspace(10, 12, n_samples)
pq1_values = np.linspace(12, 18, n_samples)
```

where there are 3 samples for PQ\_0.p0 between [10, 12] and 3 samples for PQ\_1.p0 between (12, 18).

We can use a for loop to set the load values and calculate power flow for each point.

Suppose we want to retrieve the voltage magnitude for each case, we use v\_results the voltage results. Results that are not saved will be discarded.

```
v_results = np.zeros((ss.Bus.n, n_samples ** 2))
idx = 0

for ii in pq0_values:
    ss.PQ.alter("p0", "PQ_0", ii)
    for jj in pq1_values:

        ss.PQ.alter("p0", "PQ_1", jj)

        ss.PFlow.run()
        v_results[:, idx] = ss.dae.y[ss.Bus.v.a]

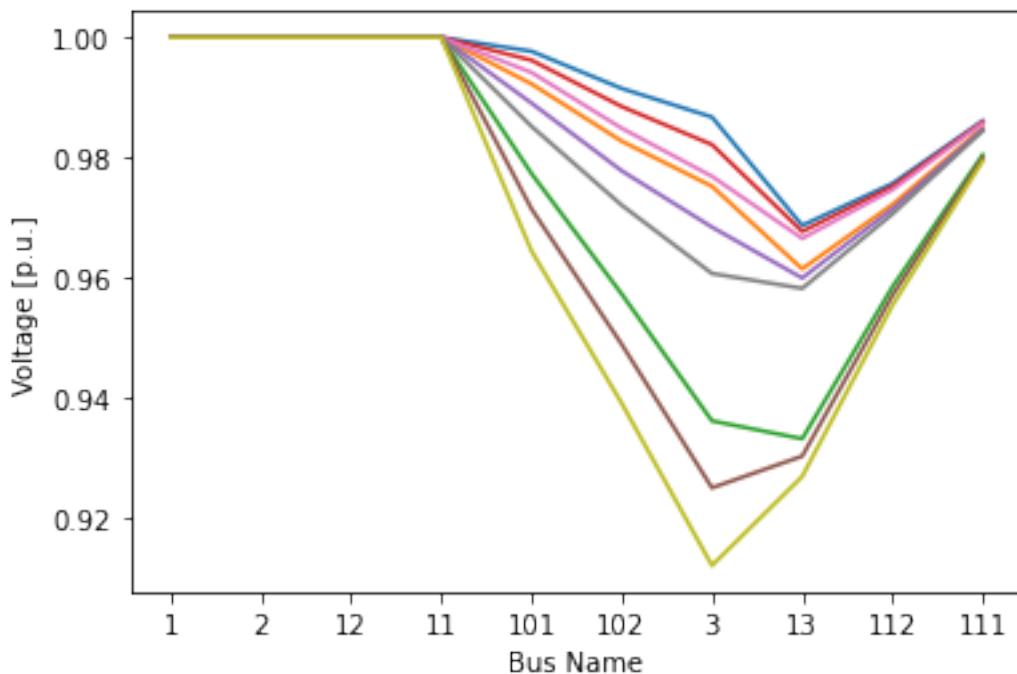
    idx += 1
```

Let's plot the results.

```
lines = plt.plot(v_results)

xl = plt.xlabel('Bus Name')
yl = plt.ylabel('Voltage [p.u.]')

tk = plt.xticks(np.arange(ss.Bus.n), ss.Bus.name.v)
```



One should be aware that the for-loop based approach is single-threaded. It does not take advantage of multi-core processors.

If the total number of scenarios are huge, one should refer to Example 7 to save all scenarios to excel files and use multi-processing.

## 2.7.2 Batch Time-Domain Simulation

The next example shows how to run batch time-domain simulations for different events.

Suppose we want to create one scenario for each line trip event, which is actuated through Toggle. For the same system, we want to add Toggles for each line, run the simulation, and save results.

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

We use `andes.load()` with `setup=False` to load the test case.

**It is important to note that one must pass `setup=False` so that adding Toggles can be allowed.**

```
ss = andes.load(kundur, setup=False)
```

The idxes of all available lines to trip are in `ss.Line.idx.v`:

```
idxes = ss.Line.idx.v
```

```
ss.Toggle.as_df()
```

	idx	u	name	model	dev	t
uid	0	1	1	Toggle_1	Line	Line_8 2

We use `ss.add()` to add two Toggles for each line at 1 second and 1.1 seconds to simulate a line opening and closing. `ss.add()` takes a model name, "Toggle", as the positional argument, and a dictionary for the Toggle device parameters.

A note for this particular test case is that `kundur_full.xls` already comes with a Toggle with `idx==1`. To not to interfere with our scenarios, we need to disable it using `ss.Toggle.alter`.

After adding Toggle devices, we need to manually call `ss.setup()` to finish the data structure setup. Then, power flow and time-domain simulation can be performed.

We store the results in a dictionary where keys are the line names and values are the systems. Code is as follows.

```
results = dict()

for idx in idxes:
    ss = andes.load(kundur, setup=False)

    ss.add('Toggle', dict(model="Line", dev=idx, t=1.0))
    ss.add('Toggle', dict(model="Line", dev=idx, t=1.1))

    ss.setup() # no `ss.add()` calls are allowed after
    ↪setup()
    ss.Toggle.alter('u', 1, 0.0) # disable the existing Toggle with idx=1
    ↪(this is for the particular case)

    ss.PFlow.run()
    ss.TDS.config.tf = 5 # simulate for 5 seconds to save time
    ss.TDS.config.no_tqdm = 1 # disable progress bar printing
    ss.TDS.run()

    results[idx] = ss
```

```
<Toggle Toggle_2>: Line.Line_0 status changed to 0 at t=1.0 sec.
<Toggle Toggle_3>: Line.Line_0 status changed to 1 at t=1.1 sec.
<Toggle Toggle_2>: Line.Line_1 status changed to 0 at t=1.0 sec.
<Toggle Toggle_3>: Line.Line_1 status changed to 1 at t=1.1 sec.
<Toggle Toggle_2>: Line.Line_2 status changed to 0 at t=1.0 sec.
<Toggle Toggle_3>: Line.Line_2 status changed to 1 at t=1.1 sec.
<Toggle Toggle_2>: Line.Line_3 status changed to 0 at t=1.0 sec.
<Toggle Toggle_3>: Line.Line_3 status changed to 1 at t=1.1 sec.
<Toggle Toggle_2>: Line.Line_4 status changed to 0 at t=1.0 sec.
<Toggle Toggle_3>: Line.Line_4 status changed to 1 at t=1.1 sec.
```

(continues on next page)

(continued from previous page)

```
<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=1.1 sec.
<Toggle Toggle_2>: Line.Line_6 status changed to 0 at t=1.0 sec.
<Toggle Toggle_3>: Line.Line_6 status changed to 1 at t=1.1 sec.
<Toggle Toggle_2>: Line.Line_7 status changed to 0 at t=1.0 sec.
<Toggle Toggle_3>: Line.Line_7 status changed to 1 at t=1.1 sec.
<Toggle Toggle_2>: Line.Line_8 status changed to 0 at t=1.0 sec.
<Toggle Toggle_3>: Line.Line_8 status changed to 1 at t=1.1 sec.
<Toggle Toggle_2>: Line.Line_9 status changed to 0 at t=1.0 sec.
<Toggle Toggle_3>: Line.Line_9 status changed to 1 at t=1.1 sec.
<Toggle Toggle_2>: Line.Line_10 status changed to 0 at t=1.0 sec.
<Toggle Toggle_3>: Line.Line_10 status changed to 1 at t=1.1 sec.
```

Time step reduced to zero. Convergence is not likely.  
 Simulation terminated at t=1.1001 s.

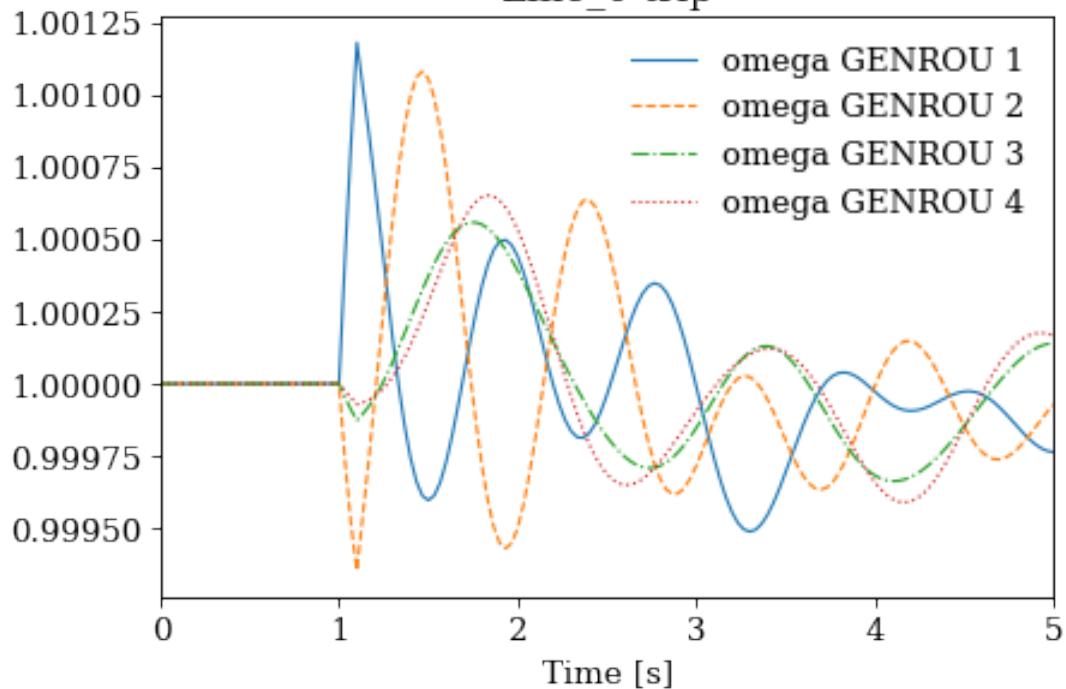
```
<Toggle Toggle_2>: Line.Line_11 status changed to 0 at t=1.0 sec.
<Toggle Toggle_3>: Line.Line_11 status changed to 1 at t=1.1 sec.
<Toggle Toggle_2>: Line.Line_12 status changed to 0 at t=1.0 sec.
<Toggle Toggle_3>: Line.Line_12 status changed to 1 at t=1.1 sec.
<Toggle Toggle_2>: Line.Line_13 status changed to 0 at t=1.0 sec.
<Toggle Toggle_3>: Line.Line_13 status changed to 1 at t=1.1 sec.
<Toggle Toggle_2>: Line.Line_14 status changed to 0 at t=1.0 sec.
<Toggle Toggle_3>: Line.Line_14 status changed to 1 at t=1.1 sec.
```

Jacobian matrix is singular.  
 Suspect diagonal elements: [55, 61]  
 NaN found in solution. Convergence is not likely  
 Simulation terminated at t=1.5808 s.

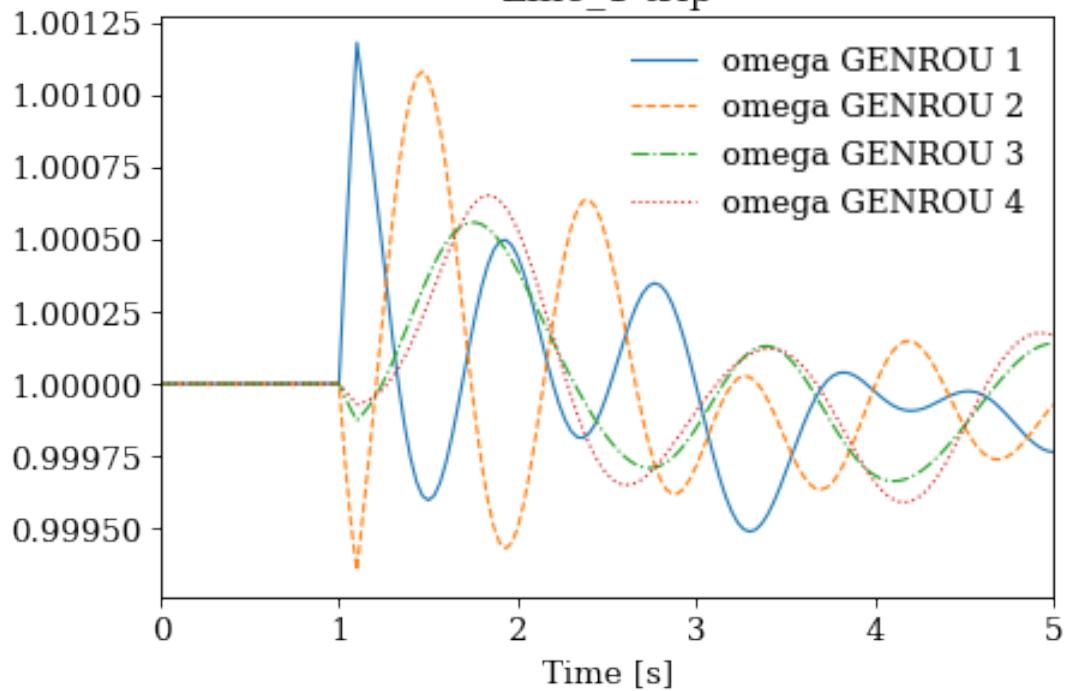
Not all cases will solve due to system instability. For the converged cases, one can export the data or plot results following Example 1.

```
for idx, ss in results.items():
    ss.TDS.plt.plot(ss.GENROU.omega, title=f'{idx} trip', latex=False, dpi=80)
```

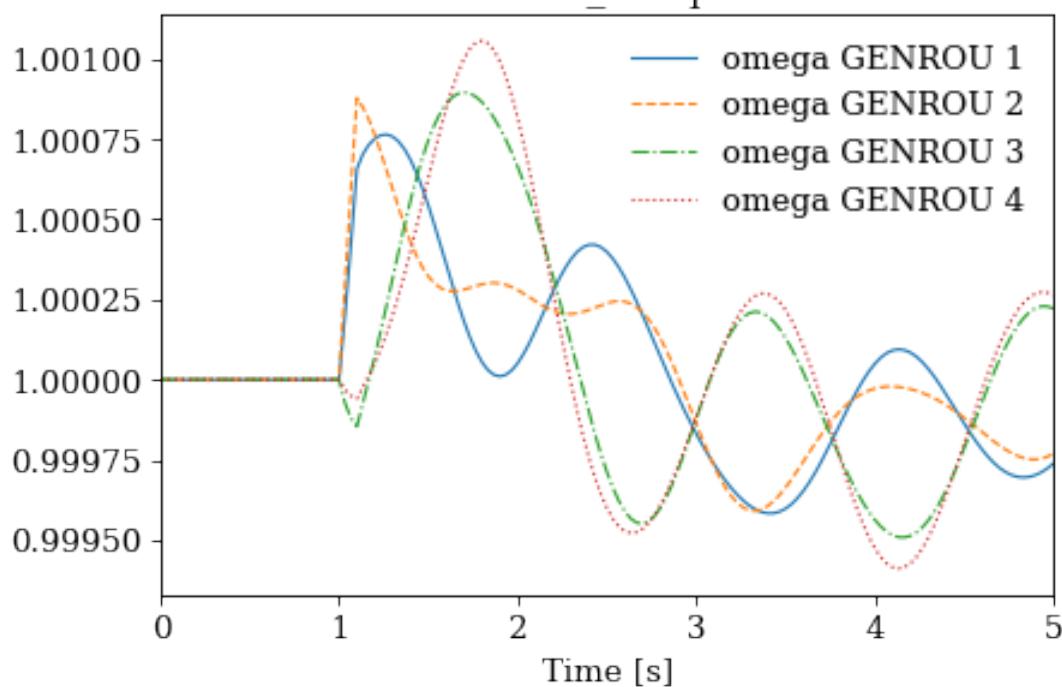
Line\_0 trip



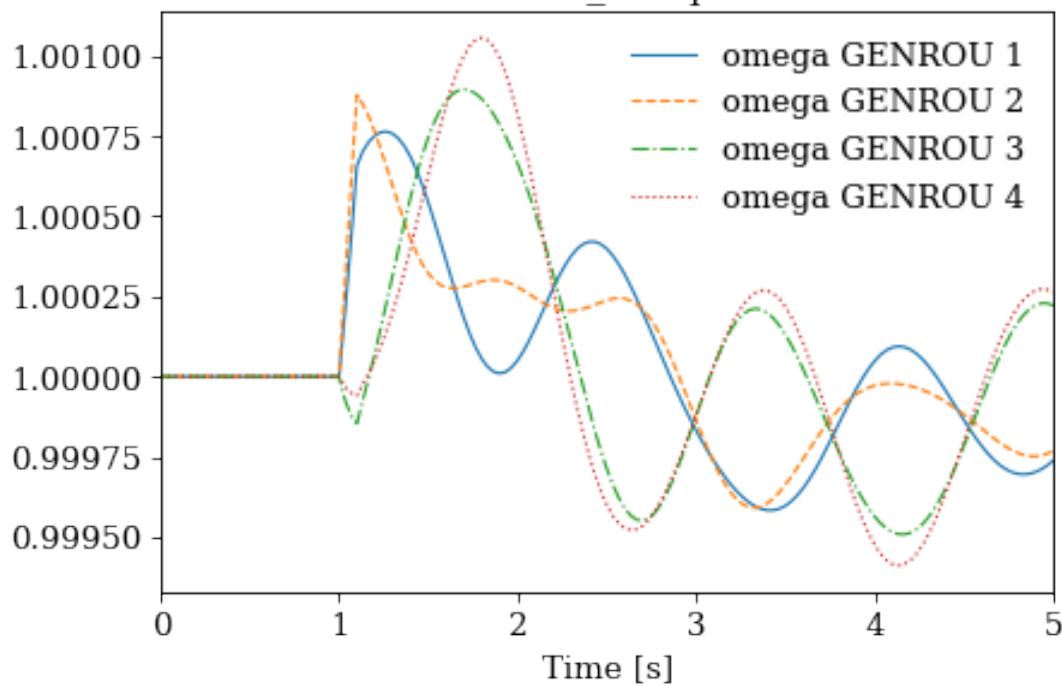
Line\_1 trip



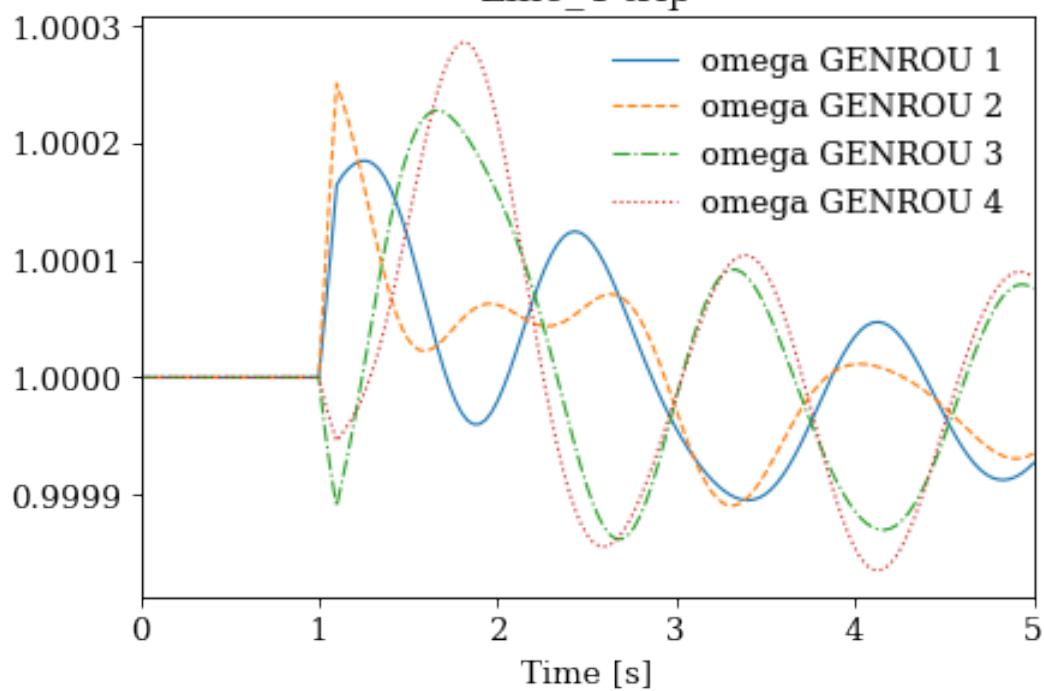
Line\_2 trip



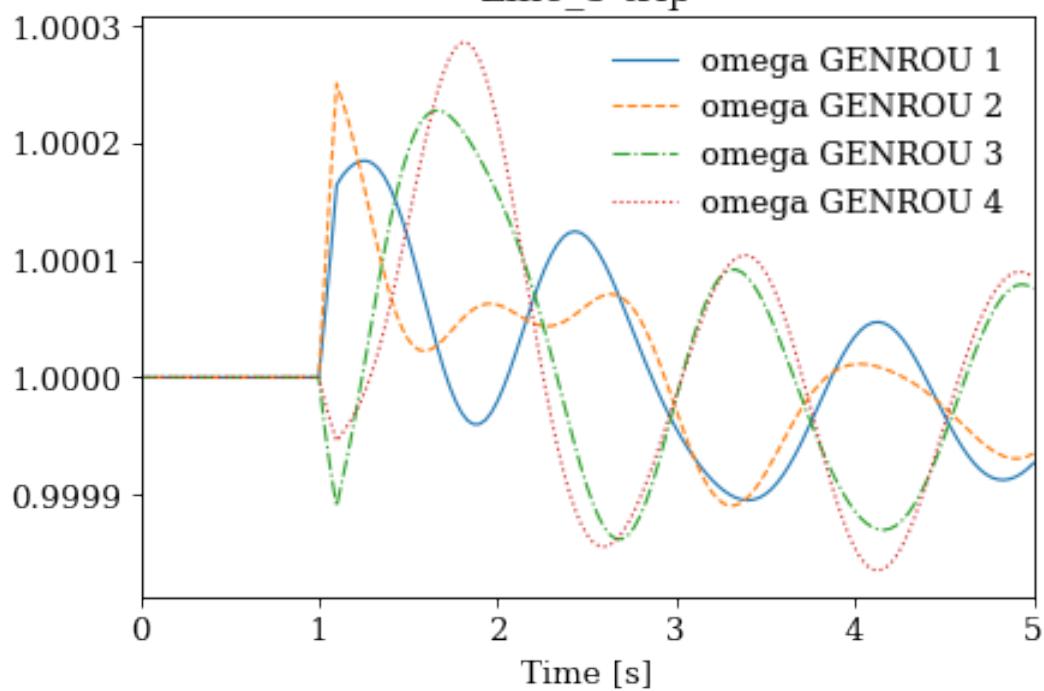
Line\_3 trip

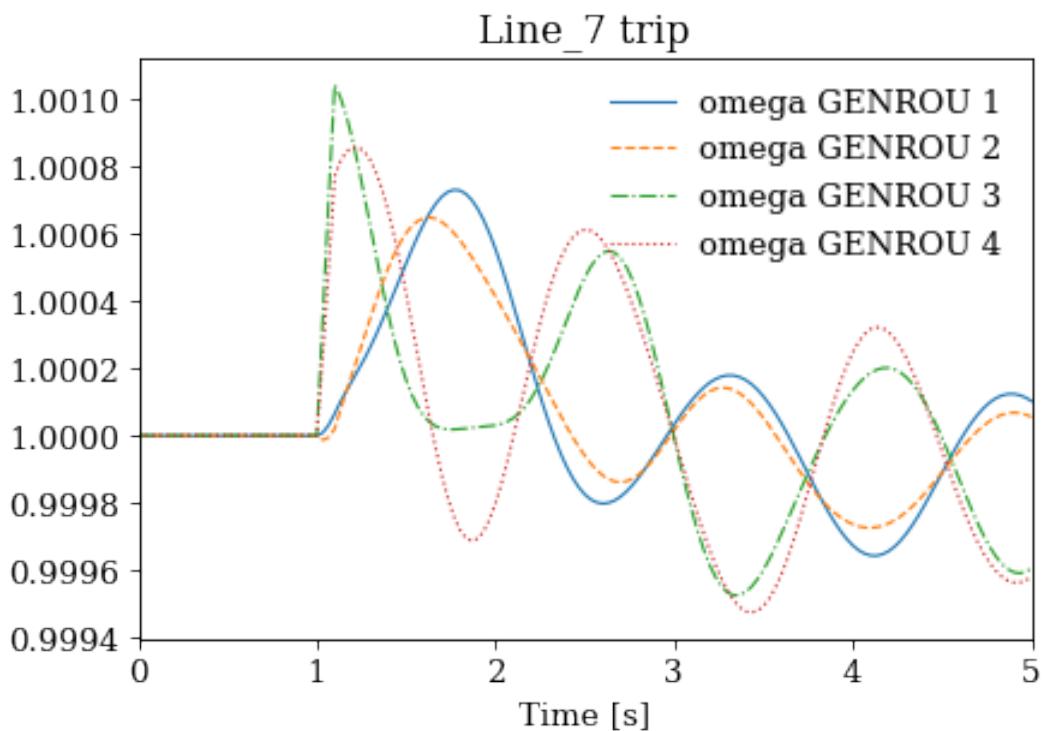
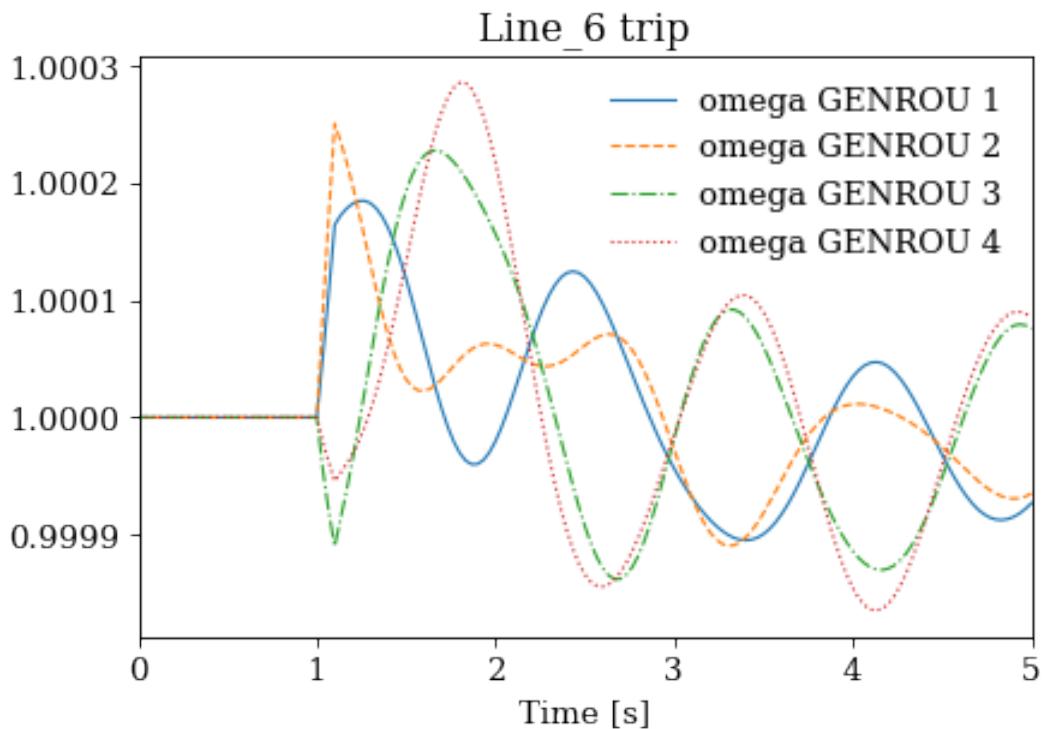


Line\_4 trip

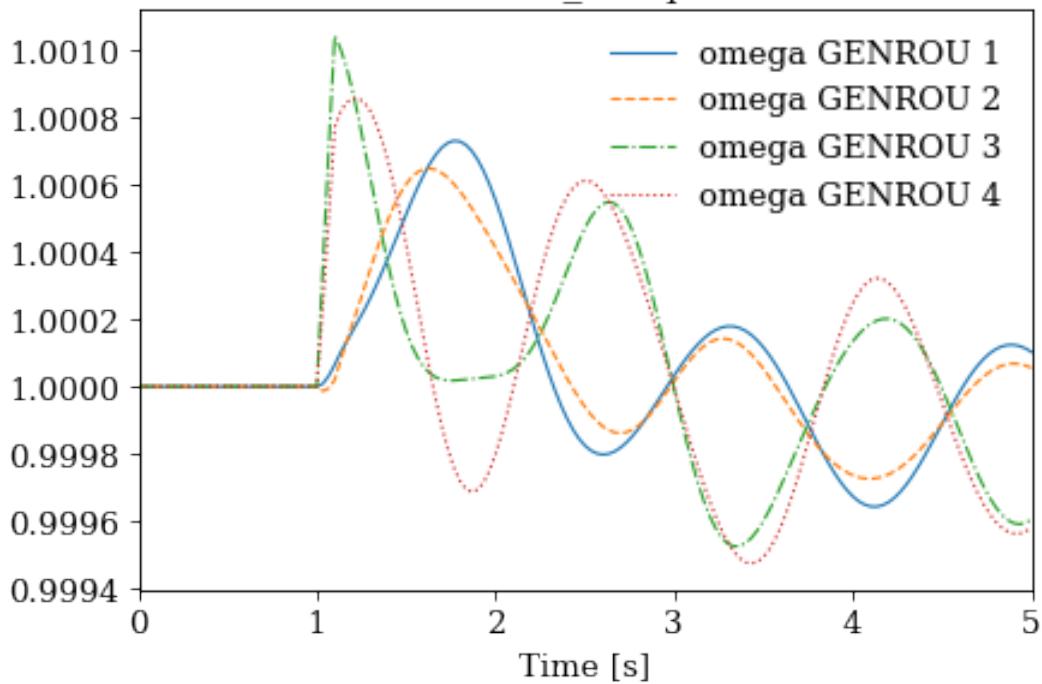


Line\_5 trip

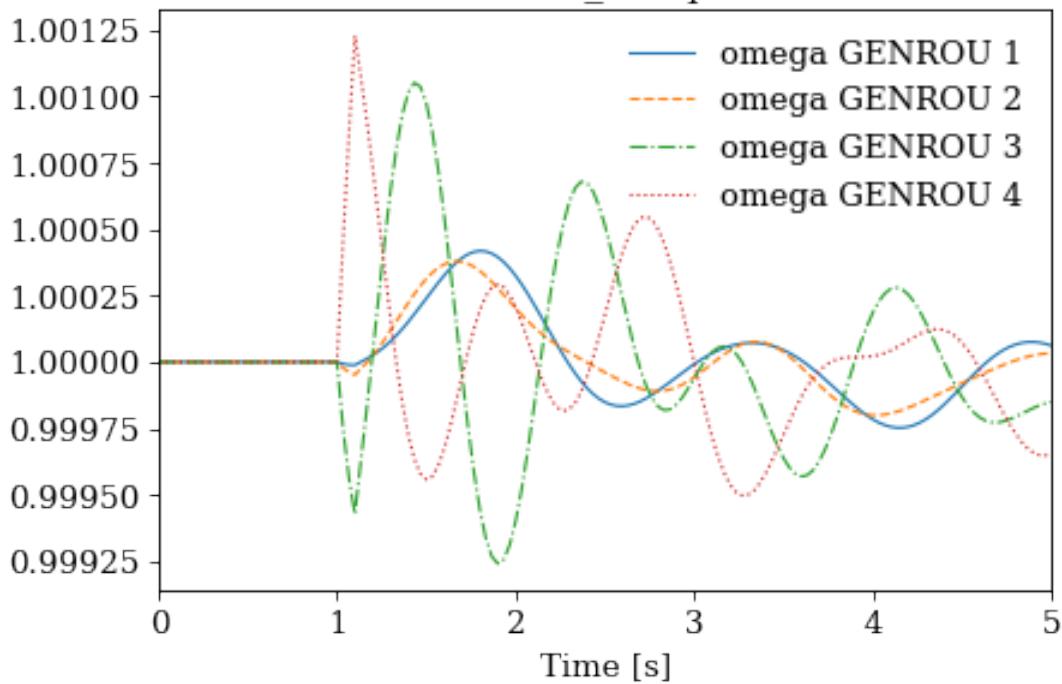




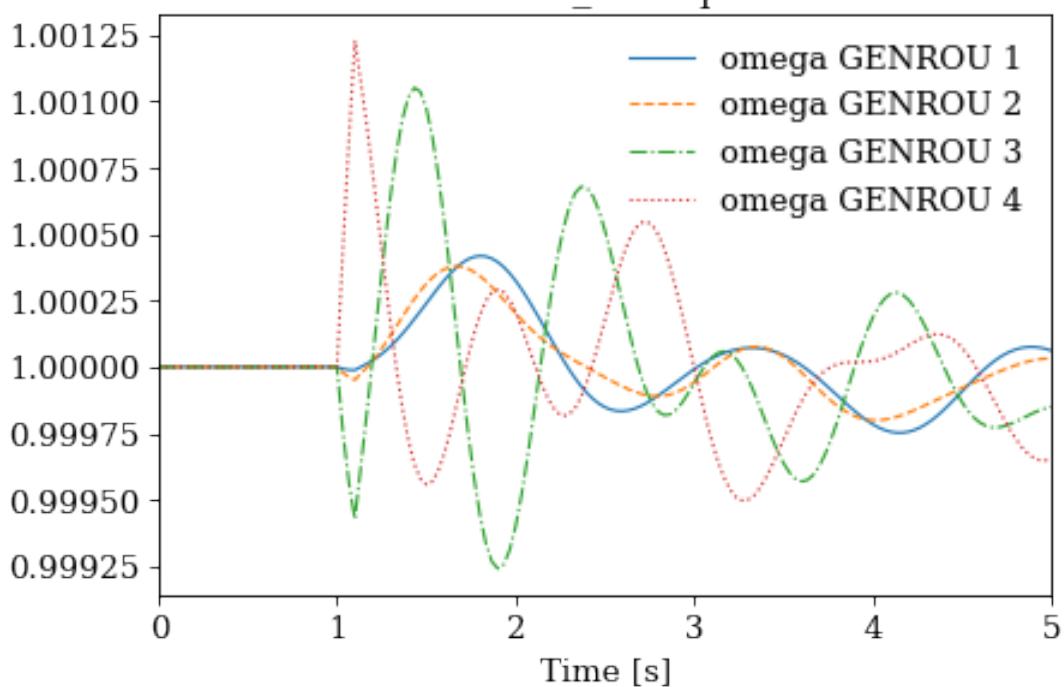
Line\_8 trip



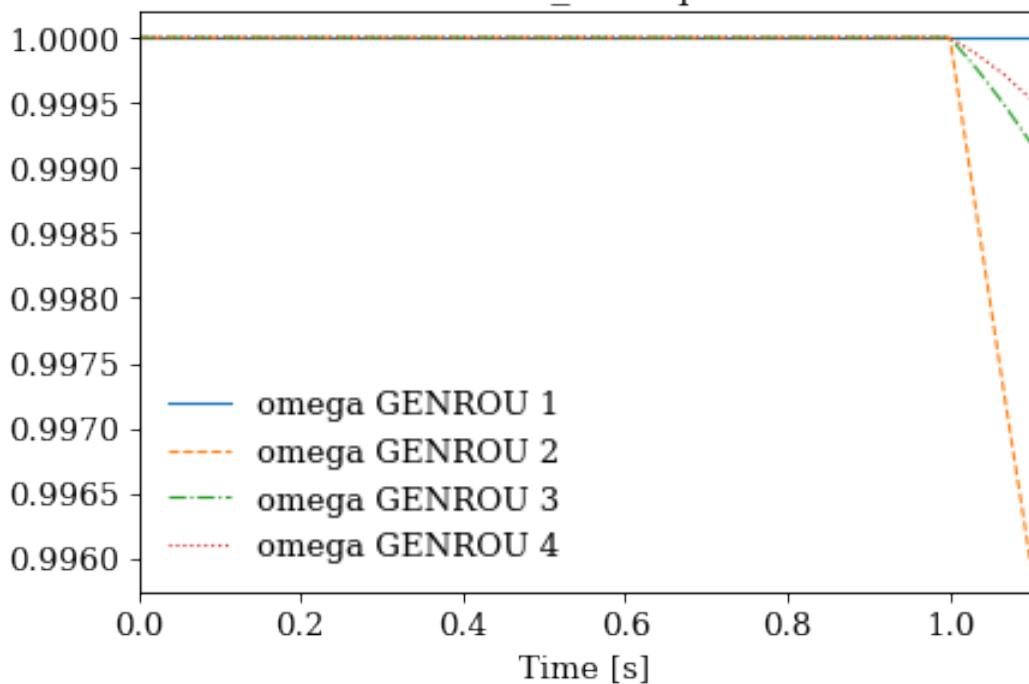
Line\_9 trip



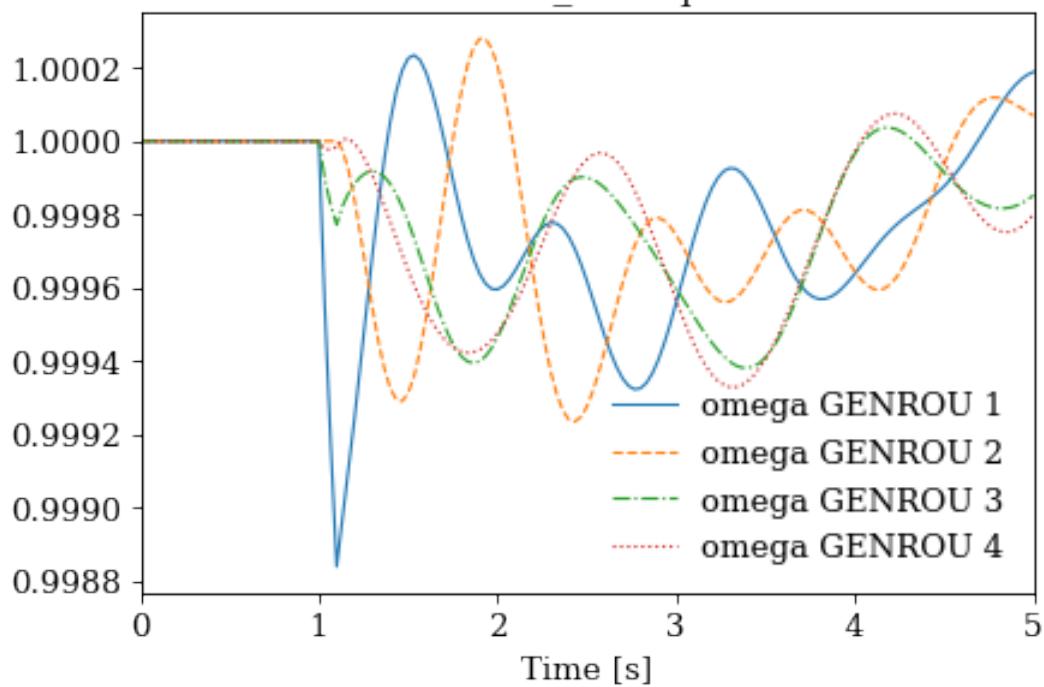
Line\_10 trip



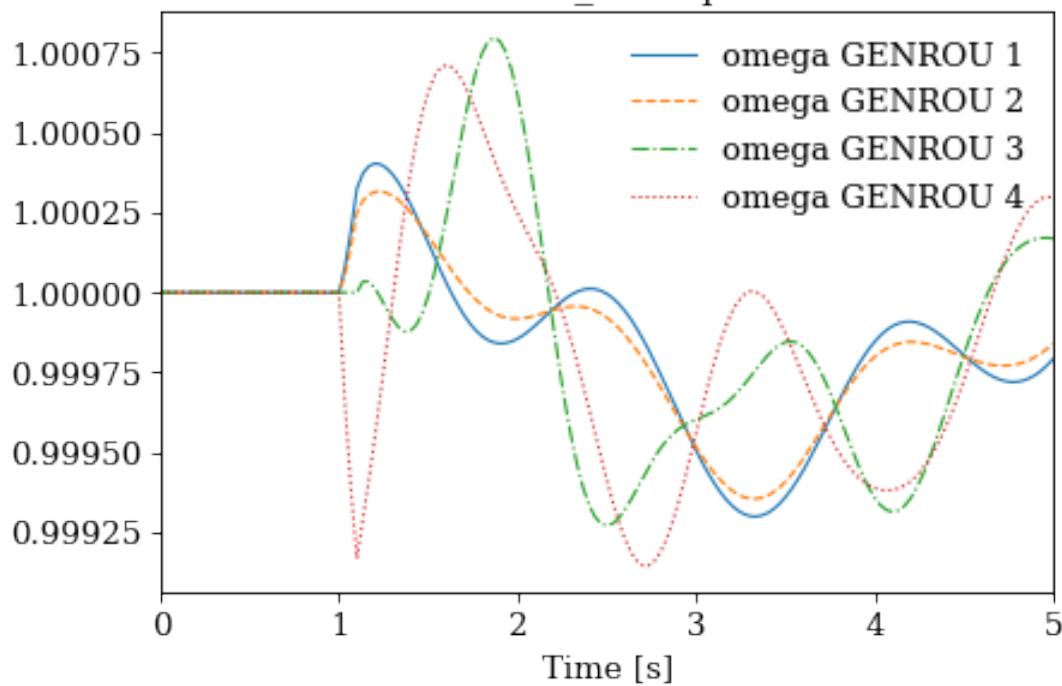
Line\_11 trip

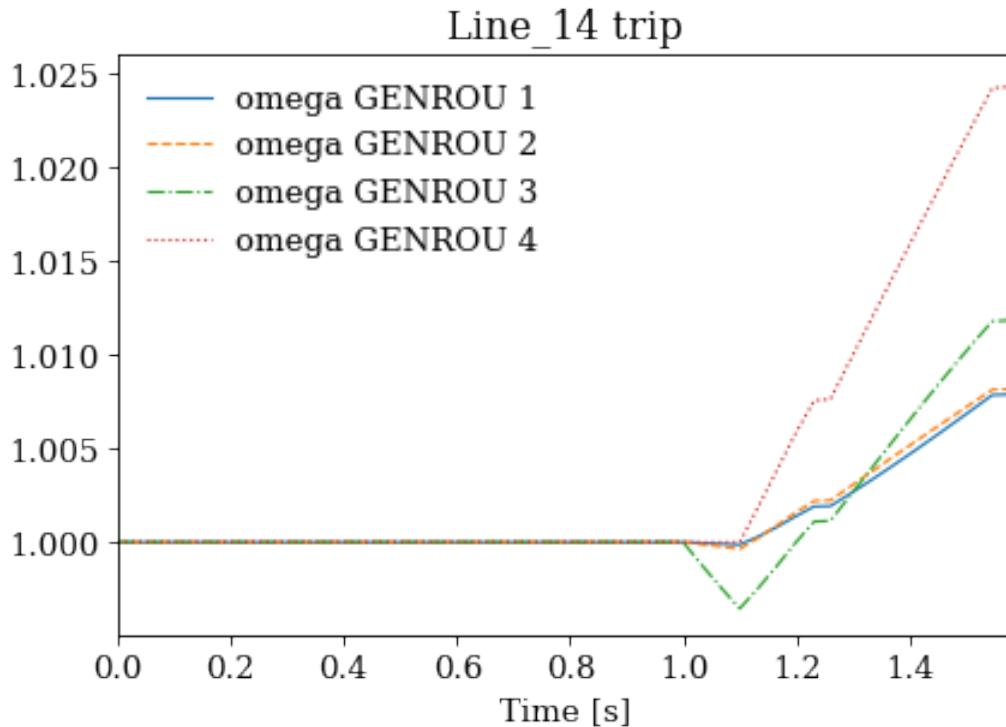


Line\_12 trip



Line\_13 trip





```
!rm -rf batch_cases/
```

## 2.8 Changing Setpoints

This notebook shows an example of changing the generator setpoints in a time-domain simulation. Data in this example is trivial, but the example can be retrofitted for scenarios such as economic dispatch incorporation or reinforcement learning.

Steps are the following:

1. Initialize a system by running the power flow,
2. Set the first simulation stop time in `TDS.config.tf`,
3. Run the simulation,
4. Update the setpoints,
5. Set the new simulation stop time and repeat from 3 until the end.

## 2.8.1 Step 1: Case Setup

```
import andes
from andes.utils import get_case
```

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

```
Working directory: "/home/hacui/repos/andes/examples"
> Loaded config from file "/home/hacui/.andes/andes.rc"
> Loaded generated Python code in "/home/hacui/.andes/pycode".
Parsing input file "/home/hacui/repos/andes/andes/cases/kundur/kundur_full.
.xlsx"...
Input file parsed in 0.2635 seconds.
System internal structure set up in 0.0315 seconds.
-> System connectivity check results:
    No islanded bus detected.
    System is interconnected.
    Each island has a slack bus correctly defined and enabled.

-> Power flow calculation
    Numba: Off
    Sparse solver: KLU
    Solution method: NR method
Power flow initialized in 0.0087 seconds.
0: |F(x)| = 14.9282832
1: |F(x)| = 3.608627841
2: |F(x)| = 0.1701107882
3: |F(x)| = 0.002038626956
4: |F(x)| = 3.745104027e-07
Converged in 5 iterations in 0.0108 seconds.
Initialization for dynamics completed in 0.0395 seconds.
Initialization was successful.
Report saved to "kundur_full_out.txt" in 0.0023 seconds.
```

```
-> Single process finished in 0.5215 seconds.
```

```
# disable the Toggle in this case
ss.Toggle.alter('u', 1, 0)
```

## 2.8.2 Step 2: Set the First Stop Time

```
# simulate to t=1 sec

# specify the first stop in `ss.TDS.config.tf`
ss.TDS.config.tf = 1
```

## 2.8.3 Step 3: Run Simulation

```
ss.TDS.run()
```

```
-> Time Domain Simulation Summary:
Sparse Solver: KLU
Simulation time: 0.0-1 s.
Fixed step size: h=33.33 ms. Shrink if not converged.
```

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

```
Simulation completed in 0.0245 seconds.
Outputs to "kundur_full_out.lst" and "kundur_full_out.npz".
Outputs written in 0.0039 seconds.
```

```
True
```

## 2.8.4 Step 4. Apply the auxiliary power setpoints to TGOV1.paux0.v

First, let's check the equations of TGOV1. ss.TGOV1.paux0 is associated with equation  $0 = \text{paux} - \text{paux0}$ , in which paux is added to the power input equation.

```
print(ss.TGOV1.doc())
```

```
Model <TGOV1> in Group <TurbineGov>
TGOV1 turbine governor model.
```

```
Implements the PSS/E TGOV1 model without deadband.
```

Parameters

Name	Description	Default	Unit	Properties
idx	unique device idx			
u	connection status	1	bool	
name	device name			
syn	Synchronous generator idx			mandatory, unique
Tn	Turbine power rating. Equal to `Sn` if not provided.		MVA	

(continues on next page)

(continued from previous page)

wref0	Base speed reference	1	p.u.	
R	Speed regulation gain (mach. base   default)	0.050	p.u.	ipower
VMAX	Maximum valve position	1.200	p.u.	power
VMIN	Minimum valve position	0	p.u.	power
T1	Valve time constant	0.100		
T2	Lead-lag lead time constant	0.200		
T3	Lead-lag lag time constant	10		
Dt	Turbine damping coefficient	0		power
Sg	Rated power from generator	0	MVA	
ug	Generator connection status	0	bool	
Vn	Rated voltage from generator	0	kV	

## Variables

Name	Type	Description	Unit	Properties
LAG_y	State	State in lag TF		v_str
LL_x	State	State in lead-lag		v_str
omega	ExtState	Generator speed	p.u.	
paux	Algeb	Auxiliary power input		v_str
pout	Algeb	Turbine final output power		v_str
wref	Algeb	Speed reference variable		v_str
pref	Algeb	Reference power input		v_str
wd	Algeb	Generator speed deviation	p.u.	v_str
pd	Algeb	Pref plus speed deviation times gain	p.u.	v_str
LL_y	Algeb	Output of lead-lag		v_str
tm	ExtAlgeb	Mechanical power interface to SynGen		

## Initialization Equations

Name	Type	Initial Value
LAG_y	State	pd * 1 / 1
LL_x	State	LAG_y
omega	ExtState	
paux	Algeb	paux0
pout	Algeb	ue * tm0
wref	Algeb	wref0
pref	Algeb	tm0 * R
wd	Algeb	0
pd	Algeb	ue * tm0
LL_y	Algeb	LAG_y
tm	ExtAlgeb	

## Differential Equations

Name	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
LAG_y	State	1 * pd - 1 * LAG_y	T1
LL_x	State	(LAG_y - LL_x)	T3
omega	ExtState		

(continues on next page)

(continued from previous page)

## Algebraic Equations

Name	Type	RHS of Equation "0 = g(x, y)"
paux	Algeb	paux0 - paux
pout	Algeb	ue * (LL_y - Dt * wd) - pout
wref	Algeb	wref0 - wref
pref	Algeb	pref0 * R - pref
wd	Algeb	ue * (omega - wref) - wd
pd	Algeb	ue*(- wd + pref + paux) * gain - pd
LL_y	Algeb	1 * T2 * (LAG_y - LL_x) + 1 * LL_x * T3 - LL_y * T3 +   LL_LT1_z1 * LL_LT2_z1 * (LL_y - 1 * LL_x)
tm	ExtAlgeb	ue * (pout - tm0)

## Services

Name	Equation	Type
ue	u * ug	ConstService
pref0	tm0	ConstService
paux0	0	ConstService
gain	ue/R	ConstService

## Discretes

Name	Type	Info
LAG_lim	AntiWindup	Limiter in Lag
LL_LT1	LessThan	
LL_LT2	LessThan	

## Blocks

Name	Type	Info
LAG	LagAntiWindup	
LL	LeadLag	

## Config Fields in [TGOV1]

Option	Value	Info	Acceptable 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)

ss.TGOV1.paux0.v

```
array([0., 0., 0., 0.])
```

```
# look up the original values of TGOV1 make sure they are as expected  
ss.TGOV1.paux0.v
```

```
array([0., 0., 0., 0.])
```

```
# MUST use in-place assignments.  
# Here, we increase the setpoint of the 0-th generator  
  
# method 1: use in-place assignment again  
  
ss.TGOV1.paux0.v[0] = 0.05  
  
# method 2: use ``ss.TGOV1.alter()``  
  
# ss.TGOV1.alter('paux0', 1, 0.05)
```

```
ss.TGOV1.paux0.v
```

```
array([0.05, 0. , 0. , 0. ])
```

Continue to simulate to 2 seconds.

```
ss.TDS.config.tf = 2
```

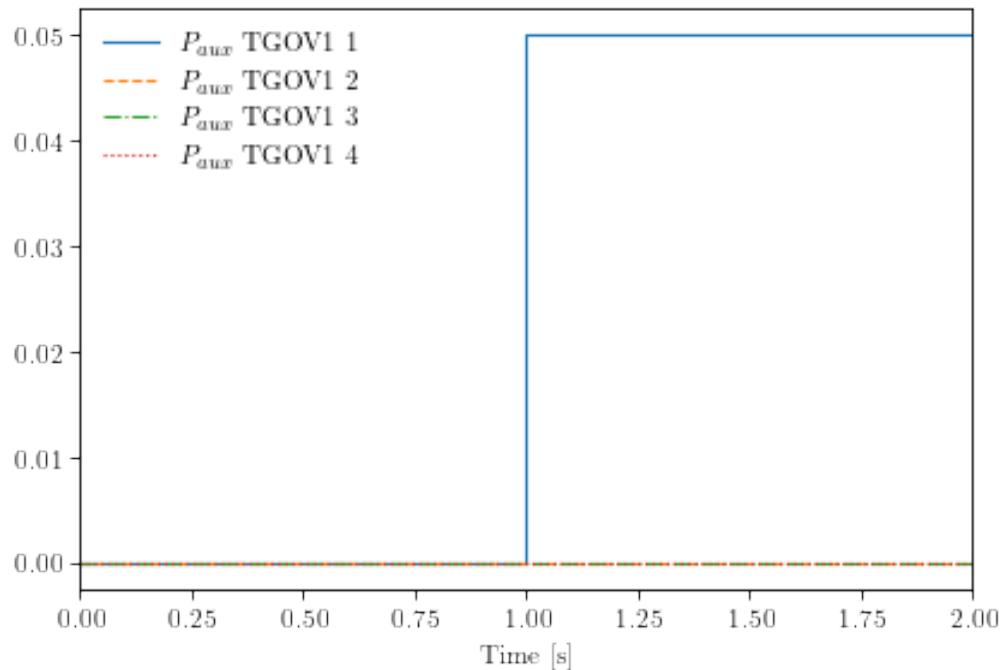
```
ss.TDS.run()
```

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

```
Simulation completed in 0.0916 seconds.  
Outputs to "kundur_full_out.lst" and "kundur_full_out.npz".  
Outputs written in 0.0032 seconds.
```

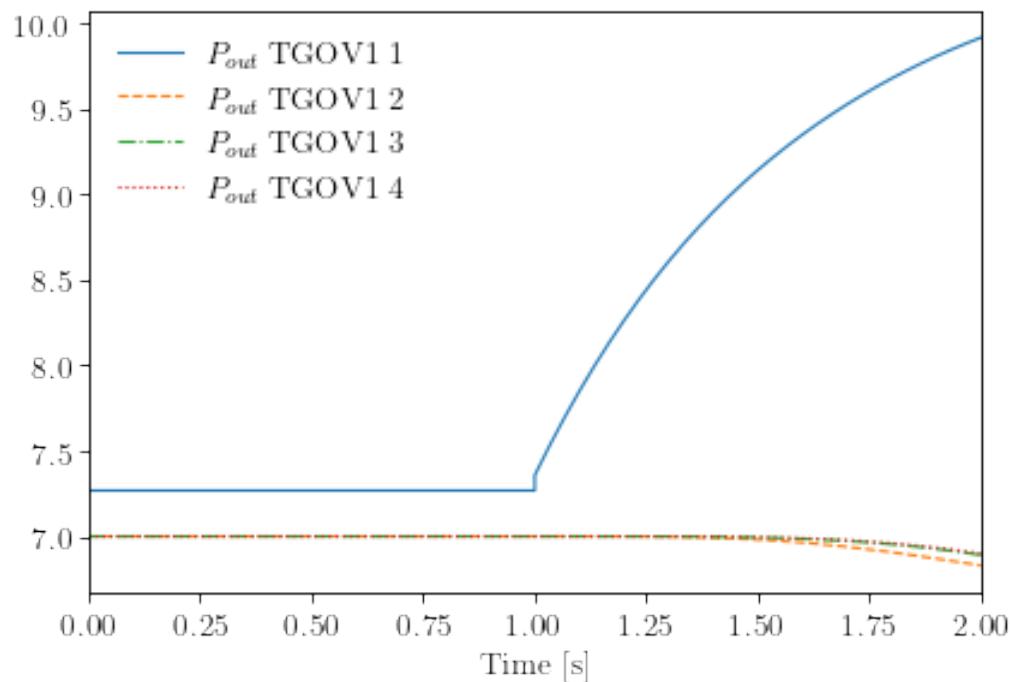
```
True
```

```
ss.TDS.plotter.plot(ss.TGOV1.paux)
```



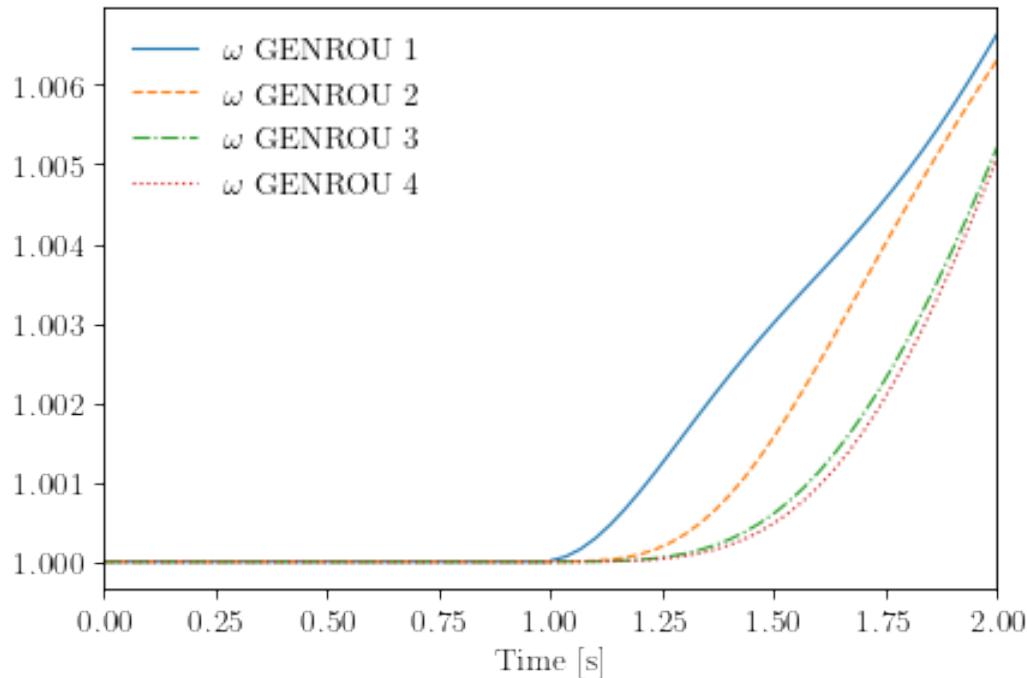
```
(<Figure size 480x320 with 1 Axes>, <AxesSubplot:xlabel='Time [s]'>)
```

```
ss.TDS.plotter.plot(ss.TGOV1.pout)
```



```
(<Figure size 480x320 with 1 Axes>, <AxesSubplot:xlabel='Time [s]'>)
```

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



```
(<Figure size 480x320 with 1 Axes>, <AxesSubplot:xlabel='Time [s]'>)
```

## 2.8.5 Step 5: Set Another New Setpoints and New Ending TIme.

In this example, we clear the auxiliary power previously set to TGOV1.paux0.v

```
# method 1: use in-place assignment again
ss.TGOV1.paux0.v[0] = 0.

# method 2: use ``ss.TGOV1.alter()``
# ss.TGOV1.alter('paux0', 1, 0)

# set the new ending time to 10 sec.
ss.TDS.config.tf = 10
```

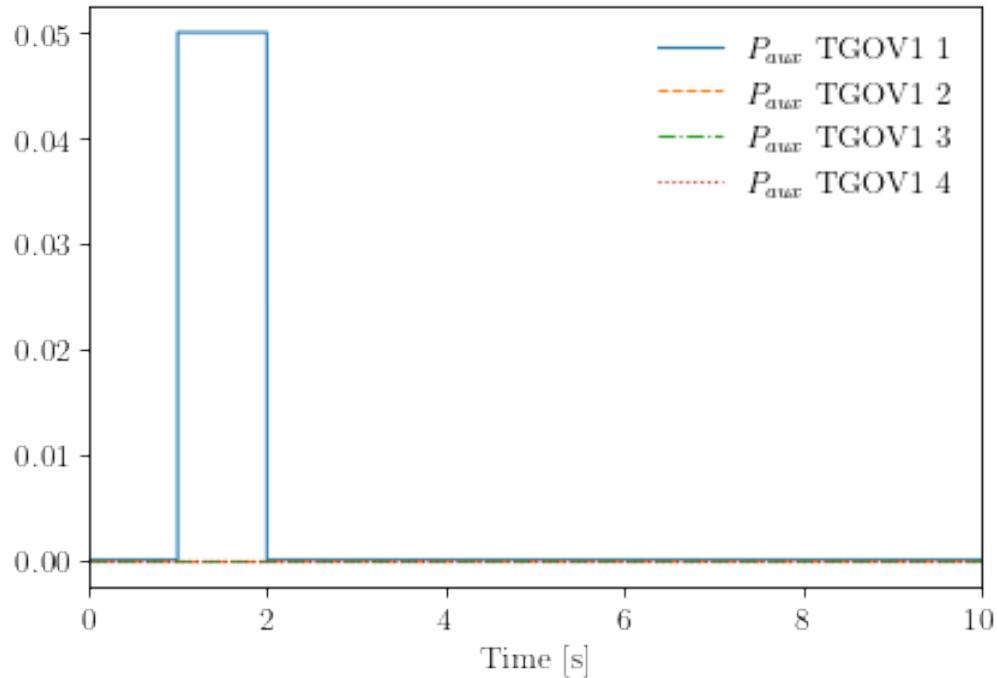
```
ss.TDS.run()
```

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

```
Simulation completed in 0.6068 seconds.
Outputs to "kundur_full_out.lst" and "kundur_full_out.npz".
Outputs written in 0.0275 seconds.
```

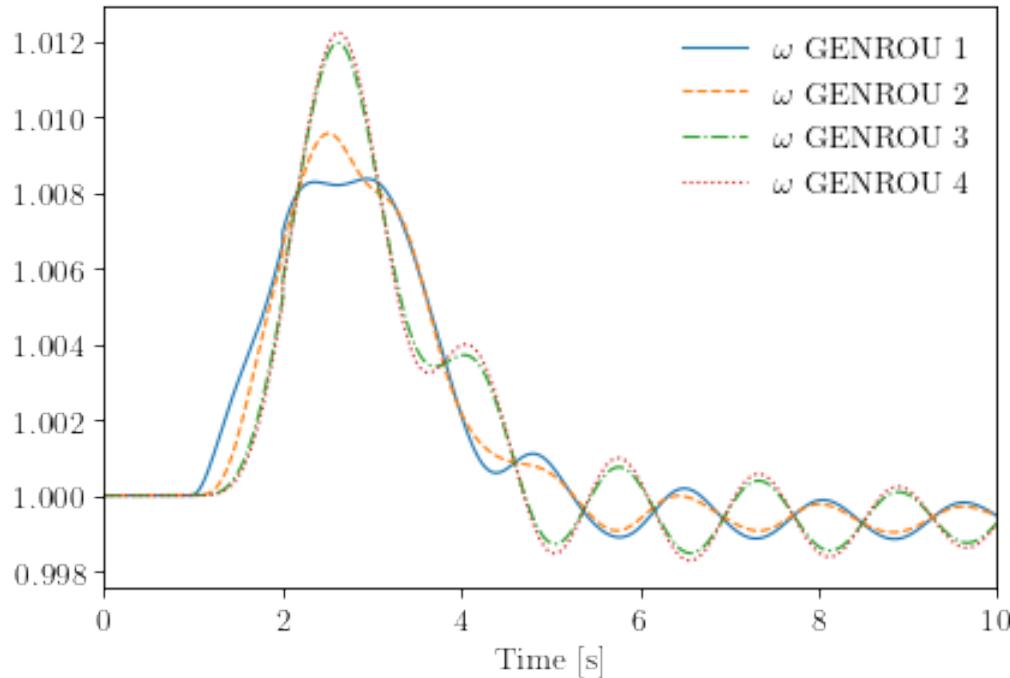
```
True
```

```
ss.TDS.plotter.plot(ss.TGOV1.paux)
```



```
(<Figure size 480x320 with 1 Axes>, <AxesSubplot:xlabel='Time [s]'>)
```

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



```
(<Figure size 480x320 with 1 Axes>, <AxesSubplot:xlabel='Time [s]'>)
```

```
!andes misc -C
```

```
"/home/hacui/repos/andes/examples/kundur_full_out.npz" removed.  
"/home/hacui/repos/andes/examples/kundur_full_out.txt" removed.  
"/home/hacui/repos/andes/examples/kundur_full_out.lst" removed.
```

## 2.9 Load Frequency Control

This examples shows (1) how to trip a generator, and (2) how to drive frequency back by load shedding.

```
import andes  
import numpy as np  
  
andes.config_logger(stream_level=20)
```

## 2.9.1 Tripping a Generator in the IEEE 14-Bus System

```
# using the IEEE 14-bus model as an example.
# The example here contains a variety of models: generators, exciters, ↵
# turbine governors, and PSS
# To speed up, one can remove unneeded ones, e.g., PSS

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

```
# use `andes.load` to load the test system
# Need to set `setup=False` to be able to add new Toggles that turns off ↵
# generators.

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

```
Working directory: "/home/hacui/repos/andes/examples"
> Loaded config from file "/home/hacui/.andes/andes.rc"
> Loaded generated Python code in "/home/hacui/.andes/pycode".
Parsing input file "/home/hacui/repos/andes/andes/cases/ieee14/ieee14.raw"...
IEEE 14 BUS TEST CASE
03/06/14 CONTO 100.0 1962 W
Input file parsed in 0.0119 seconds.
Parsing additional file "/home/hacui/repos/andes/andes/cases/ieee14/ieee14.dyr"...
Addfile parsed in 0.2965 seconds.
```

```
# Add a Toggle that disconnects `GENROU_2` at t=1 s

ss.add("Toggle", dict(model='SynGen', dev="GENROU_2", t=1.0))
```

```
'Toggle_3'
```

```
# Call setup manually

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>
System internal structure set up in 0.0750 seconds.
```

```
True
```

```
# double check that Toggles are set up correctly
# Check `u` of the Toggles - the first two line switches are disabled, and ↵
# the generator trip is enabled

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 existing line switches
# The IEEE 14-bus system contains predefined line switches. Disabling them to
# study generator trip only.

ss.Toggle.u.v[[0, 1]] = 0
```

```
# calculate power flow

# use constant power model for PQ (we will come back to this later)

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

# turn off under-voltage PQ-to-Z conversion
ss.PQ.pq2z = 0

ss.PFlow.run()
```

```
-> System connectivity check results:
No islanded bus detected.
System is interconnected.
Each island has a slack bus correctly defined and enabled.

-> Power flow calculation
    Numba: Off
    Sparse solver: KLU
    Solution method: NR method
Power flow initialized in 0.0101 seconds.
0: |F(x)| = 0.5605182134
1: |F(x)| = 0.006202200332
2: |F(x)| = 5.819382824e-06
3: |F(x)| = 6.96508129e-12
Converged in 4 iterations in 0.0111 seconds.
Initialization for dynamics completed in 0.0575 seconds.
Initialization was successful.
Report saved to "ieee14_out.txt" in 0.0083 seconds.
```

True

```
# set the first simulation stop and run it

ss.TDS.config.tf = 20
```

(continues on next page)

(continued from previous page)

```
ss.TDS.config.criteria = 0 # temporarily turn off stability criteria based
→on angle separation
ss.TDS.run()
```

-> Time Domain Simulation Summary:  
 Sparse Solver: KLU  
 Simulation time: 0.0-20 s.  
 Fixed step size: h=33.33 ms. Shrink if not converged.

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

<Toggle Toggle\_3>: SynGen.GENROU\_2 status changed to 0 at t=1.0 sec.

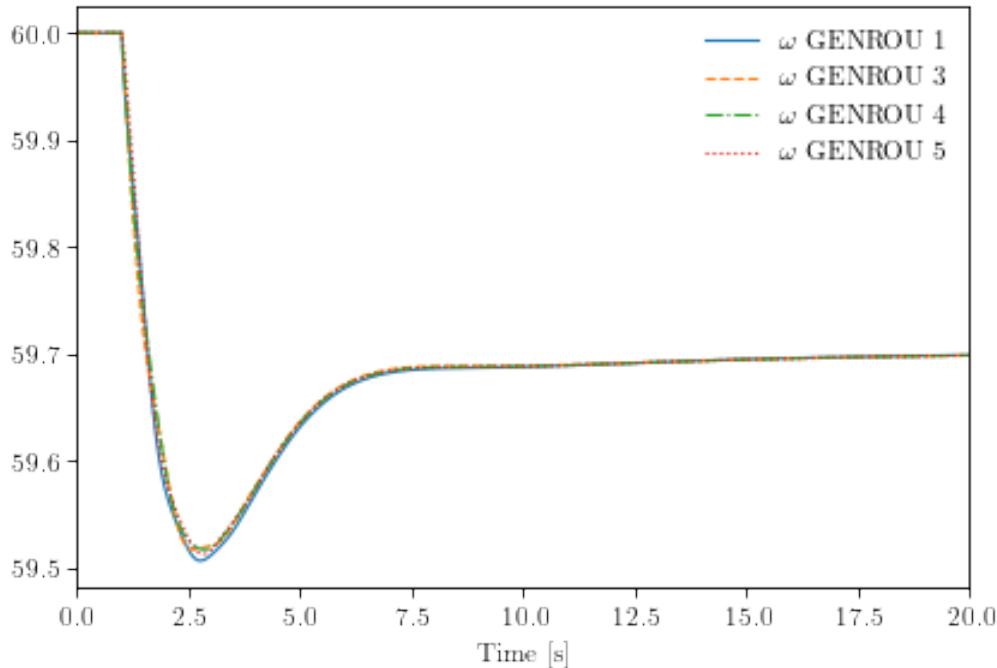
Simulation completed in 2.9748 seconds.  
 Outputs to "ieee14\_out.lst" and "ieee14\_out.npz".  
 Outputs written in 0.0421 seconds.

True

```
# Show the frequency response of online generators

# Refer to `plot` documentation by using `help(ss.TDS.plt.plot)` and `help(ss.
→TDS.plt.plot_data)`
ss.TDS.load_plotter()

ss.TDS.plt.plot(ss.GENROU.omega,
                 a=(0, 2, 3, 4),
                 ytimes=60,
                 )
```



```
(<Figure size 480x320 with 1 Axes>, <AxesSubplot:xlabel='Time [s]'>)
```

## 2.9.2 Adjusting Load to Compensate for the Generation Loss

Check the power of the lost generator by inspecting the power flow inputs:

```
ss.PV.as_df()
```

	idx	u	name	Sn	Vn	bus	busr	p0	q0	pmax	pmin	qmax	\
uid													
0	2	0.0	2	100.0	69.0	2	None	0.40	0.15	0.5	0.1	0.15	
1	3	0.0	3	100.0	69.0	3	None	0.40	0.15	0.5	0.1	0.15	
2	4	0.0	4	100.0	138.0	6	None	0.30	0.10	0.5	0.1	0.10	
3	5	0.0	5	100.0	69.0	8	None	0.35	0.10	0.5	0.1	0.10	
uid	qmin	v0	vmax	vmin	ra	xs							
0	-0.40	1.03	1.4	0.6	0.0	0.13							
1	-0.10	1.01	1.4	0.6	0.0	0.13							
2	-0.06	1.03	1.4	0.6	0.0	0.12							
3	-0.06	1.03	1.4	0.6	0.0	0.12							

The tripped GENROU\_2 correspond to the first PV (GENROU\_1 corresponds to Slack). Thus, the lost active power is 0.40 pu.

Let's compensate for that by shedding 0.4 pu of active power load at t=2.0 s.

By checking the equation documentation of PQ (using `print(ss.PQ.doc())`), we can tell that the imposed active power for time-domain simulation is from `Ppf`, because we used the constant power model with `p2p`

= 1.

### Algebraic Equations

Name	Type	RHS of Equation " $0 = g(x, y)$ "
a	ExtAlgeb	$u * (\text{dae\_t} \leq 0) * (p0 * \text{vcmp\_zi} + Rlb * \text{vcmp\_zl} * v^{**2} + Rub * \text{vcmp\_zu} * v^{**2}) + u * (\text{dae\_t} > 0) * (p2p * Ppf + p2i * Ipeq * v + p2z * Req * v^{**2})$
v	ExtAlgeb	$u * (\text{dae\_t} \leq 0) * (q0 * \text{vcmp\_zi} + Xlb * \text{vcmp\_zl} * v^{**2} + Xub * \text{vcmp\_zu} * v^{**2}) + u * (\text{dae\_t} > 0) * (q2q * Qpf + q2i * Iqeq * v + q2z * Xeq * v^{**2})$

Ppf may be different from p0 specified in the data file.

```
# active power from power flow solution - make a copy
```

```
Ppf = np.array(ss.PQ.Ppf.v)
```

```
Ppf
```

```
array([0.217, 0.5 , 0.478, 0.076, 0.15 , 0.295, 0.09 , 0.035, 0.061,
       0.135, 0.2 ])
```

Reload the system and add the generator trip.

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

ss.add("Toggle", dict(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/hacui/repos/andes/examples"
> Loaded config from file "/home/hacui/.andes/andes.rc"
> Reloaded generated Python code of module "pycode".
Parsing input file "/home/hacui/repos/andes/andes/cases/ieee14/ieee14.raw"...
IEEE 14 BUS TEST CASE
03/06/14 CONTO 100.0 1962 W
Input file parsed in 0.0047 seconds.
Parsing additional file "/home/hacui/repos/andes/andes/cases/ieee14/ieee14.dyr
...<br/>
Addfile parsed in 0.1101 seconds.
IEEEEST <IEEEEST_1> added BusFreq <BusFreq_1> linked to bus <3.0>
```

(continues on next page)

(continued from previous page)

```

ST2CUT <ST2CUT_2> added BusFreq <BusFreq_2> linked to bus <1.0>
ST2CUT <ST2CUT_3> added BusFreq <BusFreq_3> linked to bus <2.0>
System internal structure set up in 0.0547 seconds.
-> System connectivity check results:
    No islanded bus detected.
    System is interconnected.
    Each island has a slack bus correctly defined and enabled.

-> Power flow calculation
    Numba: Off
    Sparse solver: KLU
    Solution method: NR method
Power flow initialized in 0.0040 seconds.
0: |F(x)| = 0.5605182134
1: |F(x)| = 0.006202200332
2: |F(x)| = 5.819382824e-06
3: |F(x)| = 6.96508129e-12
Converged in 4 iterations in 0.0065 seconds.
Initialization for dynamics completed in 0.0547 seconds.
Initialization was successful.
Report saved to "ieee14_out.txt" in 0.0028 seconds.

```

True

But let's run to 2 seconds.

```

ss.TDS.config.tf = 2.0
ss.TDS.config.criteria = 0 # temporarily turn off stability criteria based
                           ↪on angle separation

ss.TDS.run()

```

```

-> Time Domain Simulation Summary:
Sparse Solver: KLU
Simulation time: 0.0-2.0 s.
Fixed step size: h=33.33 ms. Shrink if not converged.

```

0% | 0/100 [00:00&lt;?, ?%/s]

```
<Toggle Toggle_3>: SynGen.GENROU_2 status changed to 0 at t=1.0 sec.
```

```

Simulation completed in 0.2567 seconds.
Outputs to "ieee14_out.lst" and "ieee14_out.npz".
Outputs written in 0.0043 seconds.

```

True

```
# all `Ppf` before shedding
```

(continues on next page)

(continued from previous page)

```
ss.PQ.Ppf.v
```

```
array([0.217, 0.5 , 0.478, 0.076, 0.15 , 0.295, 0.09 , 0.035, 0.061,
       0.135, 0.2 ])
```

And then apply the load shedding on buses 2, 3, 4, 5, 6, 9.

```
shed_buses = [2, 3, 4, 5, 6, 9]

# find the `idx` of the loads on these buses

pq_shed_idx = ss.PQ.find_idx(keys='bus', values=shed_buses)
pq_shed_idx
```

```
['PQ_1', 'PQ_2', 'PQ_3', 'PQ_4', 'PQ_5', 'PQ_6']
```

```
# get `Ppf` on these buses before shedding

pq_p = ss.PQ.get(src='Ppf', idx=pq_shed_idx, attr='v')
pq_p
```

```
array([0.217, 0.5 , 0.478, 0.076, 0.15 , 0.295])
```

```
pq_p_new = pq_p - 0.4 / len(shed_buses)

ss.PQ.set(src='Ppf', idx=pq_shed_idx, attr='v', value=pq_p_new)
```

```
True
```

```
# double check

ss.PQ.Ppf.v
```

```
array([0.15033333, 0.43333333, 0.41133333, 0.00933333, 0.08333333,
       0.22833333, 0.09 , 0.035 , 0.061 , 0.135 , 0.2 ])
```

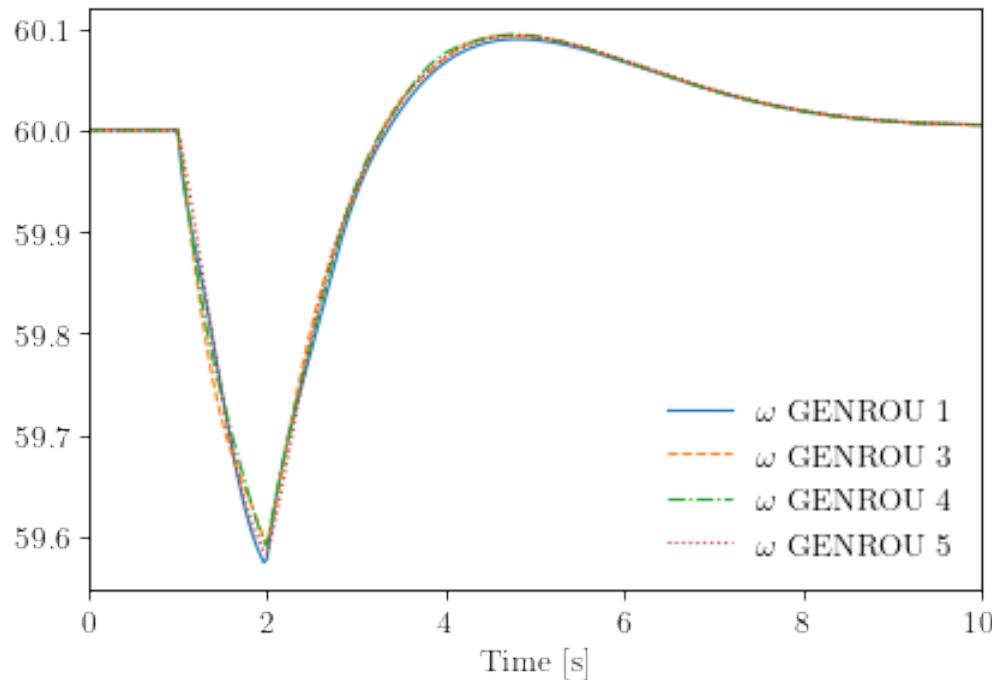
```
ss.TDS.config.tf = 10

ss.TDS.run()

ss.TDS.plt.plot(ss.GENROU.omega,
                 a=(0, 2, 3, 4),
                 ytimes=60,
                 )
```

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

```
Simulation completed in 1.2884 seconds.
Outputs to "ieee14_out.lst" and "ieee14_out.npz".
Outputs written in 0.0211 seconds.
```



```
(<Figure size 480x320 with 1 Axes>, <AxesSubplot:xlabel='Time [s]'>)
```

```
!andes misc -C
```

```
"/home/hacui/repos/andes/examples/ieee14_out.txt" removed.
"/home/hacui/repos/andes/examples/ieee14_out.lst" removed.
"/home/hacui/repos/andes/examples/ieee14_out.npz" removed.
```

The result shows the generator speed (frequency) returns to 60 Hz after load shedding.

## 2.10 Profile in Notebook

### 2.10.1 Profiling with Python CProfiler

Before getting started, this example requires the config flag `PFlow.init_tds` to be 0, which is the default value.

```
import andes
from andes.utils.paths import get_case

case_path = get_case("kundur/kundur_full.xlsx")
```

Passing `profile=True, no_output = True` to run will enable the profiler and have the results printed.

```
ss = andes.run(
    case_path, profile=True, routine="tds", no_output=True, default_
    ↪config=True
)
```

```
Working directory: "/home/hacui/repos/andes/examples"
> Loaded generated Python code in "/home/hacui/.andes/pycode".
Parsing input file "/home/hacui/repos/andes/andes/cases/kundur/kundur_full.
↪xlsx"...
Input file parsed in 0.3536 seconds.
System internal structure set up in 0.0412 seconds.
-> System connectivity check results:
  No islanded bus detected.
  System is interconnected.
  Each island has a slack bus correctly defined and enabled.

-> Power flow calculation
  Numba: Off
  Sparse solver: KLU
  Solution method: NR method
Power flow initialized in 0.0051 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 0.0045 seconds.
Initialization for dynamics completed in 0.0368 seconds.
Initialization was successful.
```

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

```
<Toggle 1>: Line.Line_8 status changed to 0 at t=2.0 sec.
```

```
Simulation to t=20.00 sec completed in 0.9577 seconds.
```

```
2126948 function calls (2100484 primitive calls) in 1.634 seconds

Ordered by: cumulative time
List reduced from 5556 to 40 due to restriction <40>

      ncalls  tottime  percall  cumtime  percall filename:lineno(function)
          1    0.006    0.006    1.065    1.065 /home/hacui/repos/andes/andes/
↪routines/tds.py:326(run)
      603    0.001    0.000    0.891    0.001 /home/hacui/repos/andes/andes/
↪routines/tds.py:521(item_step)
      603    0.051    0.000    0.890    0.001 /home/hacui/repos/andes/andes/
↪routines/daeint.py:27(step)
```

(continues on next page)

(continued from previous page)

2025	0.006	0.000	0.703	0.000	/home/hacui/repos/andes/andes/
→routines/tds.py:820(fg_update)					
10176	0.060	0.000	0.646	0.000	/home/hacui/repos/andes/andes/
→system.py:1672(call_models)					
1	0.000	0.000	0.558	0.558	/home/hacui/repos/andes/andes/
→main.py:275(load)					
1040/207	0.003	0.000	0.423	0.002	<frozen importlib._bootstrap>
→:1022(_find_and_load)					
960/129	0.002	0.000	0.420	0.003	<frozen importlib._bootstrap>
→:987(_find_and_load_unlocked)					
847/34	0.001	0.000	0.419	0.012	<frozen importlib._bootstrap_
→external>:877(exec_module)					
905/66	0.002	0.000	0.419	0.006	{built-in method builtins.exec}
1246/34	0.001	0.000	0.418	0.012	<frozen importlib._bootstrap>
→:233(_call_with_frames_removed)					
925/130	0.002	0.000	0.415	0.003	<frozen importlib._bootstrap>
→:664(_load_unlocked)					
2030	0.001	0.000	0.367	0.000	/home/hacui/repos/andes/andes/
→system.py:1047(g_update)					
1	0.000	0.000	0.355	0.355	/home/hacui/repos/andes/andes/
→io/__init__.py:115(parse)					
1	0.000	0.000	0.353	0.353	/home/hacui/repos/andes/andes/
→io/xlsx.py:88(read)					
20279	0.082	0.000	0.349	0.000	/home/hacui/repos/andes/andes/
→core/model/model.py:963(g_update)					
35	0.000	0.000	0.270	0.008	/home/hacui/repos/andes/andes/
→utils/lazyimport.py:61(__maybe_import__)					
34	0.000	0.000	0.214	0.006	/home/hacui/repos/andes/andes/
→utils/lazyimport.py:73(__getattr__)					
1	0.000	0.000	0.213	0.213	/home/hacui/mambaforge/envs/a/
→lib/python3.10/site-packages/pandas/__init__.py:1(<module>)					
1	0.000	0.000	0.169	0.169	/home/hacui/mambaforge/envs/a/
→lib/python3.10/site-packages/pandas/core/api.py:1(<module>)					
1	0.000	0.000	0.161	0.161	/home/hacui/repos/andes/andes/
→system.py:93(__init__)					
109/107	0.000	0.000	0.138	0.001	/home/hacui/mambaforge/envs/a/
→lib/python3.10/importlib/__init__.py:108(import_module)					
109/107	0.000	0.000	0.138	0.001	<frozen importlib._bootstrap>
→:1038(_gcd_import)					
1	0.000	0.000	0.122	0.122	/home/hacui/mambaforge/envs/a/
→lib/python3.10/site-packages/pandas/io/excel/_base.py:460(read_excel)					
254/196	0.000	0.000	0.108	0.001	{built-in method builtins._
→import__}					
2030	0.001	0.000	0.101	0.000	/home/hacui/repos/andes/andes/
→system.py:1033(f_update)					
1	0.000	0.000	0.100	0.100	/home/hacui/repos/andes/andes/
→system.py:1800(import_models)					
79/52	0.000	0.000	0.097	0.002	<frozen importlib._bootstrap_
→external>:1182(exec_module)					
79/52	0.082	0.001	0.097	0.002	{built-in method _imp.exec_
→dynamic}					
1636/1630	0.009	0.000	0.097	0.000	{built-in method builtins._

(continues on next page)

(continued from previous page)

```

→build_class__
    1    0.000    0.000    0.097    0.097 /home/hacui/mambaforge/envs/a/
→lib/python3.10/site-packages/pandas/io/excel/_base.py:1520(__init__)
    1    0.000    0.000    0.094    0.094 /home/hacui/mambaforge/envs/a/
→lib/python3.10/site-packages/pandas/io/excel/_openpyxl.py:534(__init__)
  20279    0.039    0.000    0.086    0.000 /home/hacui/repos/andes/andes/
→core/model/model.py:935(f_update)
  2030    0.002    0.000    0.084    0.000 /home/hacui/repos/andes/andes/
→system.py:989(l_update_eq)
    4    0.000    0.000    0.083    0.021 /home/hacui/mambaforge/envs/a/
→lib/python3.10/site-packages/pandas/compat/_optional.py:81(import_optional_
→dependency)
1864/802    0.002    0.000    0.082    0.000 <frozen importlib._bootstrap>
→:1053(_handle_fromlist)
    1    0.000    0.000    0.080    0.080 /home/hacui/mambaforge/envs/a/
→lib/python3.10/site-packages/openpyxl/__init__.py:1(<module>)
  2025    0.076    0.000    0.076    0.000 /home/hacui/.andes/pycode/
→GENROU.py:20(g_update)
    1    0.000    0.000    0.069    0.069 /home/hacui/mambaforge/envs/a/
→lib/python3.10/site-packages/pandas/core/arrays/__init__.py:1(<module>)
  20279    0.011    0.000    0.068    0.000 /home/hacui/repos/andes/andes/
→core/model/model.py:712(l_check_eq)

-> Single process finished in 1.7222 seconds.

```

## 2.10.2 Profiling with line\_profiler.

`line_profiler` provides line-based profiling results for functions.

Install with `pip install line_profiler` and restart the notebook.

```

import andes
from andes.utils.paths import get_case

case_path = get_case("kundur/kundur_full.xlsx")

```

### Profile power flow

Pass the function name to profile to the magic `%lprun`, followed by a call to the function itself or an upper-level function.

Results will be shown in a popup window.

```

%load_ext line_profiler

%lprun -f andes.routines.pflow.PFlow.run andes.run(case_path, no_output=True,_
→default_config=True)

```

```

Working directory: "/home/hacui/repos/andes/examples"
> Reloaded generated Python code of module "pycode".
Parsing input file "/home/hacui/repos/andes/andes/cases/kundur/kundur_full.
→xlsx"...
Input file parsed in 0.0498 seconds.
System internal structure set up in 0.0402 seconds.
-> System connectivity check results:
  No islanded bus detected.
  System is interconnected.
  Each island has a slack bus correctly defined and enabled.

-> Power flow calculation
  Numba: Off
  Sparse solver: KLU
  Solution method: NR method
Power flow initialized in 0.0050 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 0.0041 seconds.

```

```
-> Single process finished in 0.2605 seconds.
```

```

Timer unit: 1e-09 s

Total time: 0.0114298 s
File: /home/hacui/repos/andes/andes/routines/pflow.py
Function: run at line 211

Line #      Hits          Time  Per Hit   % Time  Line Contents
=====
211                      def run(self, **kwargs):
212                                """
213                                Solve the power flow_
→using the selected method.
214
215                                Returns
216
217                                -----
218                                bool
219                                convergence_
→status
220
221      1       1363.0    1363.0     0.0      system = self.system
222      1       581.0     581.0     0.0      if self.config.check_
→conn == 1:
223      1     1378597.0   1e+06    12.1      self.system.
→connectivity()
224
225      1     250450.0   250450.0    2.2      self.summary()

(continues on next page)

```

(continues on next page)

(continued from previous page)

```

260      1      862.0    862.0    0.0          self.y_sol =_
↳system.dae.y.copy()
261
262      1      441.0    441.0    0.0          if self.config.
↳init_tds:
263
264      1      330.0    330.0    0.0          system.TDS.
↳init()
265      1      1803.0   1803.0   0.0          if self.config.
↳report:
266
267      1      491.0    491.0    0.0          system.PFlow.
↳report()
268      1      110.0    110.0    0.0          system.exit_code = 0
↳if self.converged else 1
268      1      110.0    110.0    0.0          return self.converged

```

Alternatively, do

```

ss = andes.load(case_path, no_output=True, default_config=True)

%lprun -f ss.PFlow.run ss.PFlow.run()

```

```

Working directory: "/home/hacui/repos/andes/examples"
> Reloaded generated Python code of module "pycode".
Parsing input file "/home/hacui/repos/andes/andes/cases/kundur/kundur_full.
↳xlsx"...
Input file parsed in 0.0293 seconds.
System internal structure set up in 0.0231 seconds.
-> System connectivity check results:
  No islanded bus detected.
  System is interconnected.
  Each island has a slack bus correctly defined and enabled.

-> Power flow calculation
  Numba: Off
  Sparse solver: KLU
  Solution method: NR method
Power flow initialized in 0.0043 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 0.0040 seconds.

```

```

Timer unit: 1e-09 s

Total time: 0.0109119 s
File: /home/hacui/repos/andes/andes/routines/pflow.py
Function: run at line 211

```

(continues on next page)

(continued from previous page)

Line #	Hits	Time	Per Hit	% Time	Line Contents
211					def run(self, **kwargs):
212					"""
213					Solve the power flow
↳ using the selected method.					
214					Returns
215					-----
216					bool
217					convergence
↳ status					"""
219					
220					
221	1	972.0	972.0	0.0	system = self.system
222	1	511.0	511.0	0.0	if self.config.check_
↳ conn == 1:					self.system.
223	1	1557962.0	2e+06	14.3	
↳ connectivity()					self.summary()
224					self.init()
225	1	253977.0	253977.0	2.3	
226	1	4784035.0	5e+06	43.8	
227					if system.dae.m == 0:
228	1	451.0	451.0	0.0	logger.error(
229					
↳ "Loaded case contains no power flow element.")					system.exit_code
230					
↳ = 1					return False
231					
232					
233	1	842.0	842.0	0.0	method = self.config.
↳ method.lower()					
234					
235	1	4840.0	4840.0	0.0	t0, _ = elapsed()
236					
237					# ----- Call
↳ solution methods -----					
238	1	211.0	211.0	0.0	if method == 'nr':
239	1	4053092.0	4e+06	37.1	self.nr_solve()
240					elif method == 'nk':
241					self.newton_
↳ krylov()					
242					
243	1	5711.0	5711.0	0.1	t1, s1 = elapsed(t0)
244	1	200.0	200.0	0.0	self.exec_time = t1 -
↳ t0					
245					
246	1	241.0	241.0	0.0	if not self.
↳ converged:					
247					if abs(self.mis[-
↳ 1] - self.mis[-2]) < self.config.tol:					max_idx = np.
248					(continues on next page)

(continued from previous page)

```

→argmax(np.abs(system.dae.xy))
 249                               name =_
→system.dae.xy_name[max_idx]
 250                               logger.error(
→'Mismatch is not correctable possibly due to large load-generation_'
→imbalance.')
 251                               logger.error(
→'Largest mismatch on equation associated with <%s>', name)
 252                               else:
 253                               logger.error(
→'Power flow failed after %d iterations for "%s".', self.niter + 1, system.
→files.case)
 254
 255                               else:
 256           1     241955.0 241955.0      2.2      logger.info(
→'Converged in %d iterations in %s.', self.niter + 1, s1)
 257
 258                               # make a copy of_
→power flow solutions
 259           1     3116.0   3116.0      0.0      self.x_sol =_
→system.dae.x.copy()
 260           1     701.0    701.0      0.0      self.y_sol =_
→system.dae.y.copy()
 261
 262           1     562.0    562.0      0.0      if self.config.
→init_tds:
 263
 264           1     241.0    241.0      0.0      if self.config.
→report:
 265           1     1753.0   1753.0      0.0      system.PFlow.
→report()
 266
 267           1     441.0    441.0      0.0      system.exit_code = 0_
→if self.converged else 1
 268           1     110.0    110.0      0.0      return self.converged

```

To dig into the Newton Raphson iteration steps, profile each step instead with:

```

ss = andes.load(case_path, no_output=True, default_config=True)
%lprun -f ss.PFlow.nr_step ss.PFlow.run()

```

```

Working directory: "/home/hacui/repos/andes/examples"
> Reloaded generated Python code of module "pycode".
Parsing input file "/home/hacui/repos/andes/andes/cases/kundur/kundur_full.
→xlsx"...
Input file parsed in 0.0299 seconds.
System internal structure set up in 0.0242 seconds.
-> System connectivity check results:
  No islanded bus detected.
  System is interconnected.

```

(continues on next page)

(continued from previous page)

```

Each island has a slack bus correctly defined and enabled.

-> Power flow calculation
    Numba: Off
    Sparse solver: KLU
    Solution method: NR method
Power flow initialized in 0.0045 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 0.0040 seconds.

```

```

Timer unit: 1e-09 s

Total time: 0.00289216 s
File: /home/hacui/repos/andes/andes/routines/pflow.py
Function: nr_step at line 104

Line #      Hits          Time  Per Hit   % Time  Line Contents
=====
104                      def nr_step(self):
105                          """
106                              Solve a single_
107
108                                  iteration step using the Newton-Raphson method.
109
110                                  Returns
111
112                                  float
113
114      5       2064.0     412.8      0.1      maximum absolute_
115
116
117      5       1288374.0  257674.8     44.5      mismatch
118
119
120      5       2365.0      473.0      0.1      numerical DAE
121
122      5       1263947.0  252789.4     43.7      if self.config.
123
124
125      5       4048.0      809.6      0.1      method != 'dishonest' or (self.niter < self.config.n_factorize):
126
127      5       1263947.0  252789.4     43.7      system.j_
128
129      5       4048.0      809.6      0.1      update(self.models)
130
131      5       1263947.0  252789.4     43.7      self.solver.
132
133      5       4048.0      809.6      0.1      worker.new_A = True
134
135      5       1263947.0  252789.4     43.7      # ----- prepare_
136
137      5       4048.0      809.6      0.1      and solve linear equations

```

(continues on next page)

(continued from previous page)

```

125      5    12294.0   2458.8    0.4      self.res[:system.dae.
→n] = -system.dae.f[:]
126      5    9229.0    1845.8    0.3      self.res[system.dae.
→n:] = -system.dae.g[:]
127
128      10   15200.0   1520.0    0.5      self.A =_
→sparse([[system.dae.fx, system.dae.gx],
129      5    1994.0    398.8    0.1      _])
→[system.dae.fy, system.dae.gy]])
130
131      5    1422.0    284.4    0.0      if not self.config.
→linsolve:
132      5    95764.0   19152.8   3.3      self.inc = self.
→solver.solve(self.A, self.res)
133
134
→solver.linsolve(self.A, self.res)
135
136      5    15670.0   3134.0    0.5      system.dae.x += np.
→ravel(self.inc[:system.dae.n])
137      5    11501.0   2300.2    0.4      system.dae.y += np.
→ravel(self.inc[system.dae.n:])
138
139      # find out variables_
→associated with maximum mismatches
140      5    591.0     118.2    0.0      fmax = 0
141      5    1232.0    246.4    0.0      if system.dae.n > 0:
142                      fmax_idx = np.
→argmax(np.abs(system.dae.f))
143
144                      logger.debug(
→"Max. diff mismatch %.10g on %s", fmax, system.dae.x_name[fmax_idx])
145
146      5    28063.0   5612.6    1.0      gmax_idx = np.
→argmax(np.abs(system.dae.g))
147      5    2987.0    597.4    0.1      gmax = system.dae.
→g[gmax_idx]
148      5    87689.0   17537.8   3.0      logger.debug("Max._")
→algeb mismatch %.10g on %s", gmax, system.dae.y_name[gmax_idx])
149
150      5    6012.0    1202.4   0.2      mis = max(abs(fmax),_
→abs(gmax))
151      5    41141.0   8228.2   1.4      system.vars_to_
→models()
152
153      5    570.0     114.0    0.0      return mis

```

## Profile time-domain simulation

```
xy = ss.TDS.init()
```

```
Initialization for dynamics completed in 0.0209 seconds.  
Initialization was successful.
```

```
%lprun -f ss.TDS.item_step ss.TDS.run()
```

```
-> Time Domain Simulation Summary:  
Sparse Solver: KLU  
Simulation time: 0.0-20.0 s.  
Fixed step size: h=33.33 ms. Shrink if not converged.
```

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

```
<Toggle 1>: Line.Line_8 status changed to 0 at t=2.0 sec.
```

```
Simulation to t=20.00 sec completed in 0.9426 seconds.
```

```
Timer unit: 1e-09 s

Total time: 0.882568 s
File: /home/hacui/repos/andes/andes/routines/tds.py
Function: item_step at line 521

Line #      Hits          Time  Per Hit   % Time  Line Contents
=====
 521                      def item_step(self):
 522                          """
 523                              Integrate for the_
→step size of ``self.h`` using implicit trapezoid method.
 524
 525
 526
 527
 528
→status in ``self.converged``.
 529
 530
 531      602    882567523.0     1e+06    100.0
→step(self)
                                         """
                                         return self.method.
```

### 2.10.3 Cleanup

```
!andes misc -C
```

```
No output file found in the working directory.
```

## 2.11 MATPOWER Interface

See also the API reference for `andes.interop.matpower`.

```
import andes

from andes.interop.matpower import (start_instance,
                                      to_matpower, from_matpower)
```

### 2.11.1 Create an Octave/MATLAB instance

```
m = start_instance()
```

### 2.11.2 Convert to MATPOWER

```
ss = andes.system.example()  # load an ieee14 example case

mpc = to_matpower(m, 'mpc', ss)
```

Test if the power flow of mpc can be solved

```
m.eval("runpf(mpc)")
```

```
warning: error caught while executing handle class delete method:
'osqp_mex' undefined near line 40 column 13
OSQP Error!

MATPOWER Version 7.1, 08-Oct-2020 -- AC Power Flow (Newton)

Newton's method power flow (power balance, polar) converged in 3 iterations.

Converged in 0.01 seconds
=====
|      System Summary
|_
=====

How many?          How much?          P (MW)          Q (MVar)
```

(continues on next page)

(continued from previous page)

Buses	14	Total Gen Capacity	400.0	-112.0 to 150.0
Generators	5	On-line Capacity	400.0	-112.0 to 150.0
Committed Gens	5	Generation (actual)	226.4	49.8
Loads	11	Load	223.7	95.4
Fixed	11	Fixed	223.7	95.4
Dispatchable	0	Dispatchable	-0.0 of -0.0	-0.0
Shunts	2	Shunt (inj)	-0.0	35.3
Branches	20	Losses ( $I^2 * Z$ )	2.73	13.52
Transformers	20	Branch Charging (inj)	-	23.8
Inter-ties	0	Total Inter-tie Flow	0.0	0.0
Areas	1			

	Minimum	Maximum
Voltage Magnitude	1.010 p.u. @ bus 3	1.030 p.u. @ bus 2
Voltage Angle	-9.48 deg @ bus 14	0.00 deg @ bus 1
P Losses ( $I^2 * R$ )	-	0.50 MW @ line 1-2
Q Losses ( $I^2 * X$ )	-	2.11 MVAr @ line 8-7

=====

| Bus Data

→ |

=====

Bus #	Voltage		Generation		Load	
	Mag (pu)	Ang (deg)	P (MW)	Q (MVAr)	P (MW)	Q (MVAr)
1	1.030	0.000*	81.43	-21.62	-	-
2	1.030	-1.764	40.00	30.44	21.70	12.70
3	1.010	-3.537	40.00	12.60	50.00	25.00
4	1.011	-4.410	-	-	47.80	10.00
5	1.017	-3.843	-	-	7.60	1.60
6	1.030	-6.453	30.00	20.99	15.00	7.50
7	1.022	-4.885	-	-	-	-
8	1.030	-1.540	35.00	7.40	-	-
9	1.022	-7.246	-	-	29.50	16.60
10	1.016	-7.415	-	-	9.00	5.80
11	1.019	-7.080	-	-	3.50	1.80
12	1.017	-7.473	-	-	6.10	1.60
13	1.014	-7.721	-	-	13.50	5.80
14	1.016	-9.481	-	-	20.00	7.00
			Total:	226.43	49.80	223.70
						95.40

=====

| Branch Data

→ |

=====

Brnch #	From Bus	To Bus	From Bus P (MW)	From Bus Q (MVAr)	To Bus P (MW)	To Bus Q (MVAr)	Loss (I^2 * Z) P (MW)	Loss (I^2 * Z) Q
→ (MVAr)								
→ -								

(continues on next page)

(continued from previous page)

1	1	2	50.10	-18.36	-49.60	14.29	0.503	1.53
2	1	5	31.33	-3.26	-30.83	0.17	0.500	2.06
3	2	3	17.79	4.11	-17.63	-8.00	0.158	0.67
4	2	4	28.02	0.46	-27.58	-2.69	0.433	1.31
5	2	5	22.09	-1.13	-21.83	-1.70	0.262	0.80
6	3	4	7.63	-4.40	-7.58	3.21	0.047	0.12
7	4	5	-25.97	-5.70	26.07	5.99	0.092	0.29
8	6	11	6.91	2.37	-6.86	-2.27	0.048	0.10
9	6	12	7.93	1.32	-7.86	-1.17	0.075	0.16
10	6	13	19.15	2.76	-18.92	-2.31	0.234	0.46
11	7	9	39.12	1.46	-39.12	0.15	0.000	1.61
12	9	10	5.67	5.40	-5.65	-5.35	0.019	0.05
13	9	14	13.17	-3.85	-12.94	4.34	0.229	0.49
14	10	11	-3.35	-0.45	3.36	0.47	0.009	0.02
15	12	13	1.76	-0.43	-1.75	0.44	0.007	0.01
16	13	14	7.17	-3.93	-7.06	4.16	0.111	0.23
17	4	7	4.12	-3.76	-4.12	3.83	-0.000	0.06
18	4	9	9.22	-1.06	-9.22	1.53	0.000	0.47
19	6	5	-18.99	7.03	18.99	-6.06	0.000	0.97
20	8	7	35.00	7.40	-35.00	-5.29	0.000	2.11
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								

## 2.11.4 Add dynamic data

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'))
```

```
<andes.system.System at 0x7f8684043520>
```

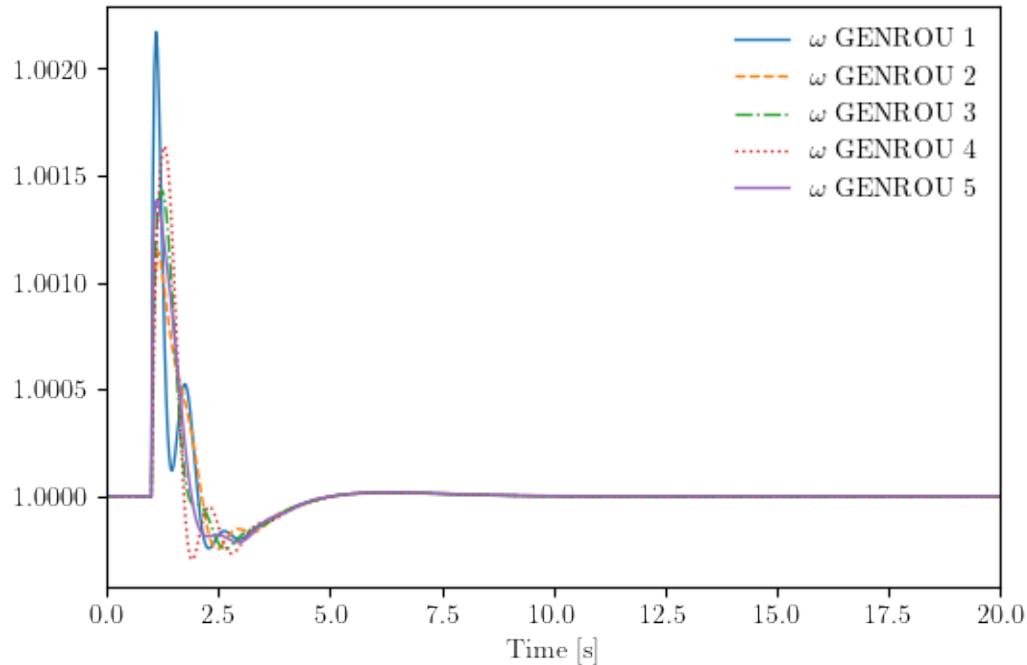
```
system.setup()
system.PFlow.run()
system.TDS.run()
```

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

```
<Fault 1>: Applying fault on Bus (idx=7) at t=1.0 sec.
<Fault 1>: Clearing fault on Bus (idx=7) at t=1.03333 sec.
```

```
True
```

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



```
(<Figure size 600x400 with 1 Axes>, <AxesSubplot:xlabel='Time [s]'>)
```

Since the data in `ieee14_dyn_only.xlsx` is stripped from `ieee14/ieee14_fault.xlsx`, the

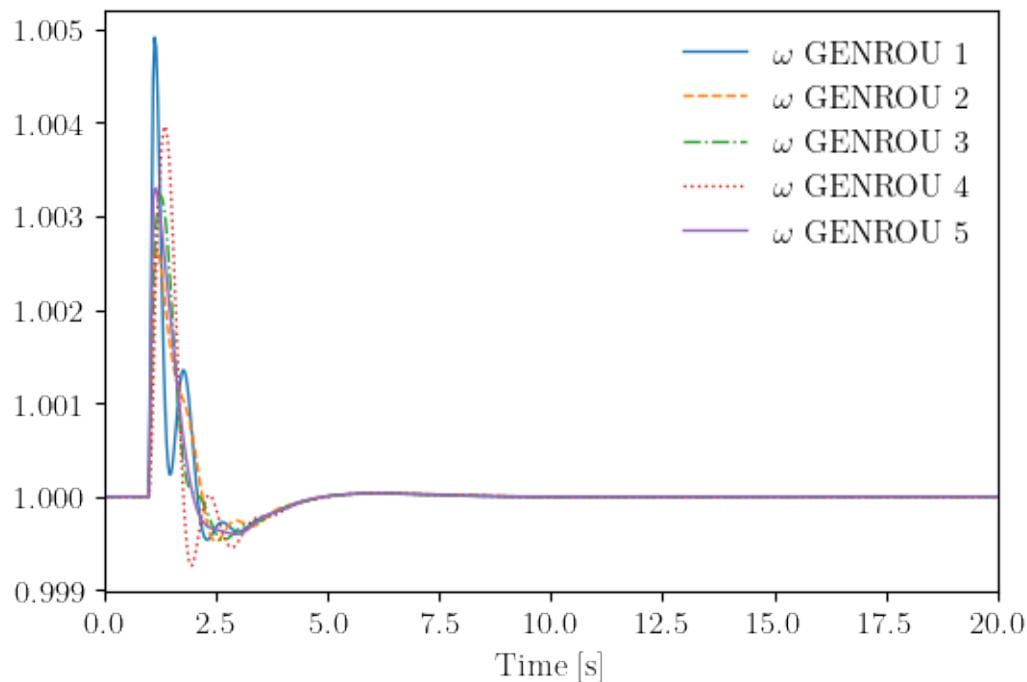
above results is identical to the simulation of the latter case:

```
ss = andes.run(andes.get_case("ieee14/ieee14_fault.xlsx"),
                routine='tds', verbose=30,
                no_output=True, default_config=True)
```

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

```
<Fault 1>: Applying fault on Bus (idx=9) at t=1.0 sec.
<Fault 1>: Clearing fault on Bus (idx=9) at t=1.1 sec.
-> Single process finished in 0.7220 seconds.
```

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



```
(<Figure size 600x400 with 1 Axes>, <AxesSubplot:xlabel='Time [s]'>)
```

## 2.12 pandapower Interface

This example (1) shows how to convert an ANDES system (`ssa`) to a pandapower network (`ssp`), (2) benchmarks the powerflow results, (3) shows how to alter `ssa` active power setpoints according to `ssp` results.

The following clarifications you might need to know:

1. This interface tracks static power flow model in ANDES: Bus, Line, PQ, Shunt, PV, and Slack. The dynamic model in ANDES is not tracked, including but not limited to TurbineGov, SynGen, and Exciter.

2. The interface converts the Slack in ANDES to gen in pandapower rather than ext\_gen.
3. **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.
4. 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.

This interface is mainly developed by Jinning Wang.

```
import andes

from andes.interop.pandapower import to_pandapower, \
    make_link_table, runopp_map, add_gencost

andes.config_logger(20)

import pandapower as pp
import numpy as np
import pandas as pd

from math import pi
```

### 2.12.1 Load case

Here we use the same ANDES system ssa to do the pandapower conversion, which leaves the ssa untouched.

This will be useful if you need to modify the ssa parameters or setpoints.

```
ssa = andes.load(andes.get_case('ieee14/ieee14_ieeet1.xlsx'),
                  setup=False,
                  no_output=True,
                  default_config=True)
ssa.Toggle.u.v = [0, 0]
ssa.setup()

ss0 = andes.load(andes.get_case('ieee14/ieee14_ieeet1.xlsx'),
                  setup=False,
                  no_output=True,
                  default_config=True)
ss0.Toggle.u.v = [0, 0]
ss0.setup()
```

```
Working directory: "/Users/jinningwang/Documents/work/andes/docs/source/
→examples"
> Loaded generated Python code in "/Users/jinningwang/.andes/pycode".
Generated code for <PQ> is stale.
Numerical code generation (rapid incremental mode) started...
```

Generating code for 1 models on 8 processes.

```
Saved generated pycode to "/Users/jinninwang/.andes/pycode"  
> Reloaded generated Python code of module "pycode".  
Generated numerical code for 1 models in 0.1328 seconds.  
Parsing input file "/Users/jinninwang/Documents/work/andes/andes/cases/  
↳ieee14/ieee14_ieeet1.xlsx"..."  
Input file parsed in 0.0441 seconds.  
System internal structure set up in 0.0236 seconds.  
Working directory: "/Users/jinninwang/Documents/work/andes/docs/source/  
↳examples"  
> Reloaded generated Python code of module "pycode".  
Generated code for <PQ> is stale.  
Numerical code generation (rapid incremental mode) started...
```

```
Generating code for 1 models on 8 processes.
```

```
Saved generated pycode to "/Users/jinninwang/.andes/pycode"  
> Reloaded generated Python code of module "pycode".  
Generated numerical code for 1 models in 0.2553 seconds.  
Parsing input file "/Users/jinninwang/Documents/work/andes/andes/cases/  
↳ieee14/ieee14_ieeet1.xlsx"..."  
Input file parsed in 0.0367 seconds.  
System internal structure set up in 0.0227 seconds.
```

```
True
```

## 2.12.2 Convert to pandapower net

Convert ADNES system ssa to pandapower net ssp.

```
ssp = to_pandapower(ss0)
```

```
-> System connectivity check results:  
No islanded bus detected.  
System is interconnected.  
Each island has a slack bus correctly defined and enabled.  
  
-> Power flow calculation  
    Numba: Off  
    Sparse solver: KLU  
    Solution method: NR method  
Power flow initialized in 0.0029 seconds.  
0: |F(x)| = 0.5605182134  
1: |F(x)| = 0.006202200332  
2: |F(x)| = 5.819382825e-06  
3: |F(x)| = 6.957087684e-12  
Converged in 4 iterations in 0.0038 seconds.  
Power flow results are consistent. Conversion is successful.
```

Add generator cost data.

```
gen_cost = np.array([
    [2, 0, 0, 3, 0.0430293, 20, 0],
    [2, 0, 0, 3, 0.25, 20, 0],
    [2, 0, 0, 3, 0.01, 40, 0],
    [2, 0, 0, 3, 0.01, 40, 0],
    [2, 0, 0, 3, 0.01, 40, 0]
])

add_gencost(ssp, gen_cost)
```

True

Inspect the pandapower net ssp.

ssp

This pandapower network includes the following parameter tables:

- bus (14 elements)
- load (11 elements)
- gen (5 elements)
- shunt (2 elements)
- line (16 elements)
- trafo (4 elements)
- poly\_cost (5 elements)

and the following results tables:

- res\_bus (14 elements)
- res\_line (16 elements)
- res\_trafo (4 elements)
- res\_load (11 elements)
- res\_shunt (2 elements)
- res\_gen (5 elements)

### 2.12.3 Comapre Power Flow Results

Run power flow of ssa.

ssa.PFlow.run()

```
-> System connectivity check results:
No islanded bus detected.
System is interconnected.
Each island has a slack bus correctly defined and enabled.

-> Power flow calculation
      Numba: Off
      Sparse solver: KLU
      Solution method: NR method
Power flow initialized in 0.0039 seconds.
0: |F(x)| = 0.5605182134
```

(continues on next page)

(continued from previous page)

```

1: |F(x)| = 0.006202200332
2: |F(x)| = 5.819382825e-06
3: |F(x)| = 6.957087684e-12
Converged in 4 iterations in 0.0045 seconds.

```

True

```

# ssa
ssa_res_gen = pd.DataFrame(columns=['name', 'p_mw', 'q_mvar', 'va_degree',
                                     'vm_pu'])
ssa_res_gen['name'] = ssa.PV.as_df()['name']
ssa_res_gen['p_mw'] = ssa.PV.p.v * ssa.config.mva
ssa_res_gen['q_mvar'] = ssa.PV.q.v * ssa.config.mva
ssa_res_gen['va_degree'] = ssa.PV.a.v * 180 / pi
ssa_res_gen['vm_pu'] = ssa.PV.v.v
ssa_res_slack = pd.DataFrame([ssa.Slack.name.v[0], ssa.Slack.p.v[0] * ssa.
                               config.mva,
                               ssa.Slack.q.v[0] * ssa.config.mva, ssa.Slack.a.
                               v[0] * 180 / pi,
                               ssa.Slack.v.v[0]]),
                               columns=ssa_res_gen.columns,
                               )
ssa_res_gen = pd.concat([ssa_res_gen, ssa_res_slack]).reset_index(drop=True)

# ssp
pp.runpp(ssp)
ssp_res_gen = pd.concat([ssp.gen['name'], ssp.res_gen], axis=1)

res_gen_concat = pd.concat([ssa_res_gen, ssp_res_gen], axis=1)

# ssa
ssa_pf_bus = ssa.Bus.as_df()[["name"]].copy()
ssa_pf_bus['v_andes'] = ssa.Bus.v.v
ssa_pf_bus['a_andes'] = ssa.Bus.a.v * 180 / pi

# ssp
ssp_pf_bus = ssa.Bus.as_df()[["name"]].copy()
ssp_pf_bus['v_pp'] = ssp.res_bus['vm_pu']
ssp_pf_bus['a_pp'] = ssp.res_bus['va_degree']

pf_bus_concat = pd.concat([ssa_pf_bus, ssp_pf_bus], axis=1)

```

## Generation

In the table below, the left half are ANDES results, and the right half are from pandapower

`res_gen_concat.round(4)`

name	p_mw	q_mvar	va_degree	vm_pu	name	p_mw	q_mvar	va_degree	vm_pu
0 2	40.0000	30.4361	-1.7641	1.03	2	40.0000	30.4361	-1.7641	
1 3	40.0000	12.5971	-3.5371	1.01	3	40.0000	12.5971	-3.5371	
2 4	30.0000	20.9866	-6.4527	1.03	4	30.0000	20.9866	-6.4527	
3 5	35.0000	7.3964	-1.5400	1.03	5	35.0000	7.3964	-1.5400	
4 1	81.4272	-21.6171	0.0000	1.03	1	81.4272	-21.6171	0.0000	

vm_pu
0 1.03
1 1.01
2 1.03
3 1.03
4 1.03

## Bus voltage and angle

Likewise, the left half are ANDES results, and the right half are from pandapower

`pf_bus_concat.round(4)`

uid	name	v_andes	a_andes	name	v_pp	a_pp
0 BUS1	1.0300	0.0000	BUS1	1.0300	0.0000	
1 BUS2	1.0300	-1.7641	BUS2	1.0300	-1.7641	
2 BUS3	1.0100	-3.5371	BUS3	1.0100	-3.5371	
3 BUS4	1.0114	-4.4098	BUS4	1.0114	-4.4098	
4 BUS5	1.0173	-3.8430	BUS5	1.0173	-3.8430	
5 BUS6	1.0300	-6.4527	BUS6	1.0300	-6.4527	
6 BUS7	1.0225	-4.8852	BUS7	1.0225	-4.8852	
7 BUS8	1.0300	-1.5400	BUS8	1.0300	-1.5400	
8 BUS9	1.0218	-7.2459	BUS9	1.0218	-7.2459	
9 BUS10	1.0155	-7.4155	BUS10	1.0155	-7.4155	
10 BUS11	1.0191	-7.0797	BUS11	1.0191	-7.0797	
11 BUS12	1.0174	-7.4730	BUS12	1.0174	-7.4730	
12 BUS13	1.0145	-7.7208	BUS13	1.0145	-7.7208	
13 BUS14	1.0163	-9.4811	BUS14	1.0163	-9.4811	

## 2.12.4 Generator dispatch based on OPF from pandapower

Prepare the link table.

```
# Asign the StaticGen with OPF, in this case, all the SynGen are GENROU
link_table = make_link_table(ssa)
```

```
link_table
```

	stg_name	stg_u	stg_idx	bus_idx	dg_idx	rg_idx	rexc_idx	syg_idx	exc_idx	\	
0		1	1.0		1	1	NaN	NaN	NaN	GENROU_1	ESST3A_2
1		2	1.0		2	2	NaN	NaN	NaN	GENROU_2	EXST1_1
2		3	1.0		3	3	NaN	NaN	NaN	GENROU_3	ESST3A_3
3		4	1.0		4	6	NaN	NaN	NaN	GENROU_4	ESST3A_4
4		5	1.0		5	8	NaN	NaN	NaN	GENROU_5	1

	gov_idx	bus_name	gammap	gammaq
0	TGOV1_1	BUS1	1.0	1.0
1	TGOV1_2	BUS2	1.0	1.0
2	TGOV1_3	BUS3	1.0	1.0
3	TGOV1_4	BUS6	1.0	1.0
4	TGOV1_5	BUS8	1.0	1.0

Note: some Jupyter notebooks may not support inline plots. You can use the following command to enable it.

```
import matplotlib
%matplotlib inline
```

```
import matplotlib
%matplotlib inline
```

Run the TDS in ADNES to 2s.

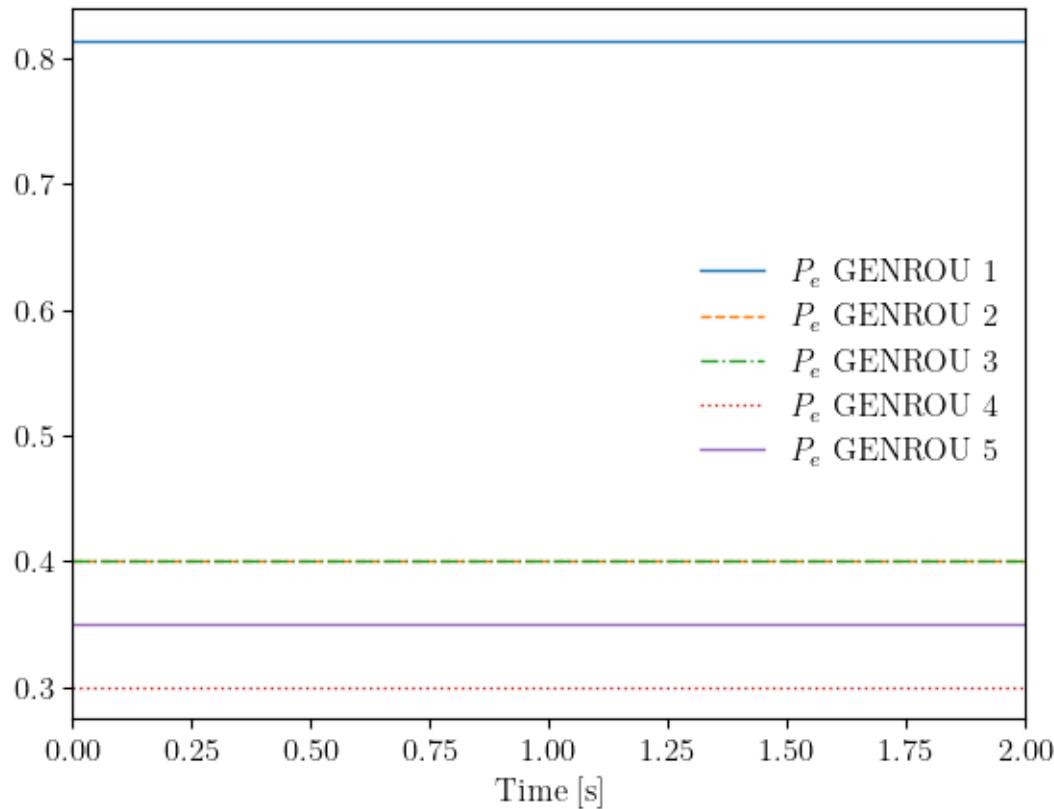
```
ssa.TDS.config.tf = 2
ssa.TDS.run()

ssa.TDS.plt.plot(ssa.GENROU.Pe)
```

```
-> Time Domain Simulation Summary:
Sparse Solver: KLU
Simulation time: 0.0-2 s.
Fixed step size: h=33.33 ms. Shrink if not converged.
Initialization for dynamics completed in 0.1282 seconds.
Initialization was successful.
```

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

```
Simulation to t=2.00 sec completed in 0.0623 seconds.
```



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

Get the OPF results from pandapower. The `ssp_res` has been converted to p.u..

```
ssp_res = runopp_map(ssp, link_table)
ssp_res
```

```
ACOPF is solved.
```

stg_idx	p	q	vm_pu	bus_idx	controllable	dg_idx	rg_idx	\
0	2	0.5	0.129107	1.096903	2	True	NaN	NaN
1	3	0.41275	0.149997	1.082551	3	True	NaN	NaN
2	4	0.34426	0.099998	1.086084	6	True	NaN	NaN
3	5	0.495878	0.085204	1.099996	8	True	NaN	NaN
4	1	0.5	-0.079016	1.100000	1	True	NaN	NaN
syg_idx	gov_idx	exc_idx						
0	GENROU_2	TGOV1_2	EXST1_1					
1	GENROU_3	TGOV1_3	ESST3A_3					
2	GENROU_4	TGOV1_4	ESST3A_4					
3	GENROU_5	TGOV1_5	1					
4	GENROU_1	TGOV1_1	ESST3A_2					

Now dispatch the results into `ssa`, where the active power setpoints are updated to `TurbinGov.pref0`.

```
ssa.gov_idx = list(ssp_res['gov_idx'][~ssp_res['gov_idx'].isna()])
ssa.TurbineGov.set(src='pref0', idx=ssa.gov_idx, attr='v', value=ssp_res['p
↪'] [~ssp_res['gov_idx'].isna()])
ssa.TurbineGov.get(src='pref0', idx=ssa.gov_idx, attr='v')
```

```
array([0.49999998, 0.41274963, 0.34426    , 0.49587786, 0.50000031])
```

Now run the TDS to 50s.

```
ssa.TDS.config.tf = 50
ssa.TDS.run()
```

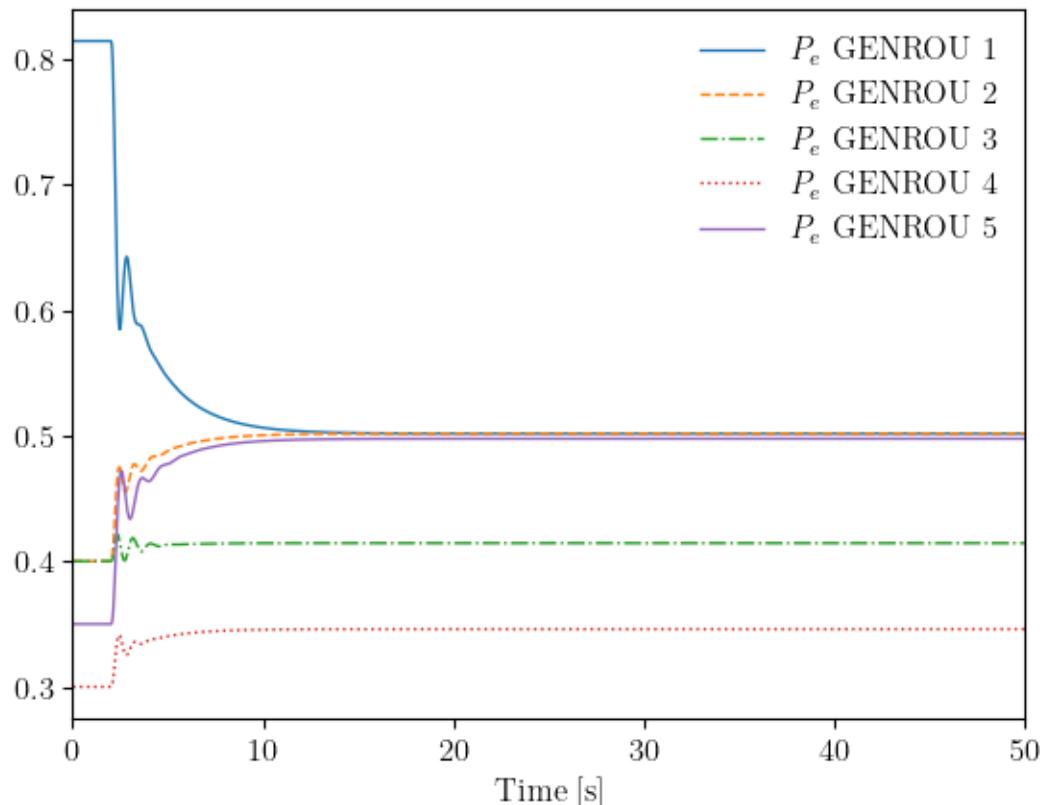
```
0% | 0/100 [00:00<?, ?%/s]
```

```
Simulation to t=50.00 sec completed in 1.9001 seconds.
```

```
True
```

We can see the outputs of GENROU are rearranged by the OPF results.

```
ssa.TDS.plt.plot(ssa.GENROU.Pe)
```



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

## 2.13 pypowsybl Interface

"PowSyBl (Power System Blocks) is an open source framework written in Java, dedicated to electrical grid modelling and simulation, licensed under the Mozilla Public License version 2.0. It is part of LF Energy, an open source foundation focused on the power systems sector, hosted within The Linux Foundation."

pypowsybl is the Python interface to PowSyBl. For more information, please visit:

<https://www.powsybl.org/pages/overview/>

ANDES provides a simple interface to pypowsybl. The main goal is to leverage pypowsybl for drawing single-line diagrams and area diagrams for systems loaded in ANDES.

### 2.13.1 Imports

```
import andes
import pypowsybl as pp

from andes.interop.pypowsybl import to_pypowsybl
```

```
ss = andes.load(andes.get_case("ieee14/ieee14_linetrip.xlsx"))
```

### 2.13.2 Conversion

Convert to a pypowsybl network:

```
n = to_pypowsybl(ss)
```

Run power flow in pypowsybl:

```
results = pp.loadflow.run_ac(n)
```

### 2.13.3 Diagrams

pypowsybl provides two functions to draw diagrams:

- `get_network_area_diagram(VOLTAGE_LEVEL_ID = None, DEPTH = 0)`
- `get_single_line_diagram(CONTAINER_ID)`

These function names are self explanatory. To draw the diagram for the whole network, do

```
n.get_network_area_diagram()
```

```
<pypowsybl.network.Svg at 0x7f1f905b33d0>
```

To draw the SLD for bus 6, do

```
n.get_single_line_diagram("VL6")
```

```
<pypowsybl.network.Svg at 0x7f1e879e2760>
```

where VL6 is the voltage level container created for Bus 6. Voltage level container is named as VL + bus idx.

To find out how to work with pypowsybl, visit its documentation at <https://pypowsybl.readthedocs.io>

## DEVELOPMENT

This chapter contains advanced topics on modeling and simulation and how they are implemented in ANDES. It aims to provide an in-depth explanation of how the ANDES framework is set up for symbolic modeling and numerical simulation. It also provides an example for interested users to implement customized DAE models.

### 3.1 System

#### 3.1.1 Overview

System is the top-level class for organizing power system models and orchestrating calculations. The full API reference of System is found at [andes.system.System](#).

#### Dynamic Imports

System dynamically imports groups, models, and routines at creation. To add new models, groups or routines, edit the corresponding file by adding entries following examples.

```
andes.system.System.import_models(self)
```

Import and instantiate models as System member attributes.

Models defined in `models/__init__.py` will be instantiated *sequentially* as attributes with the same name as the class name. In addition, all models will be stored in dictionary `System.models` with model names as keys and the corresponding instances as values.

#### Examples

`system.Bus` stores the `Bus` object, and `system.GENCLS` stores the classical generator object,

`system.models['Bus']` points the same instance as `system.Bus`.

```
andes.system.System.import_groups(self)
```

Import all groups classes defined in `models/group.py`.

Groups will be stored as instances with the name as class names. All groups will be stored to dictionary `System.groups`.

```
andes.system.System.import_routines(self)
Import routines as defined in routines/__init__.py.
```

Routines will be stored as instances with the name as class names. All routines will be stored to dictionary System.routines.

## Examples

System.PFlow is the power flow routine instance, and System.TDS and System.EIG are time-domain analysis and eigenvalue analysis routines, respectively.

## Code Generation

Under the hood, all models whose equations are provided in strings need be processed to generate executable functions for simulations. We call this process "code generation". Code generation utilizes SymPy, a symbolic toolbox, and can take up to one minute.

Code generation is automatically triggered upon the first ANDES run or whenever model changes are detected. Code generation only needs to run once unless the generated code is removed or model edits are detected. The generated code is then 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 path to store `pycode` is `HOME_DIR/.andes`, where `HOME_DIR` is one's [home directory](#).

---

**Note:** Code generation has been done if one has 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=1)
```

Generate numerical functions from symbolically defined models.

All procedures in this function must be independent of test case.

### Parameters

#### quick

[bool, optional] True to skip pretty-print generation to reduce code generation time.

#### incremental

[bool, optional] True to generate only for modified models, incrementally.

**models**

[list, OrderedDict, None] List or OrderedDict of models to prepare

**nomp**

[bool] True to disable multiprocessing

**Warning:** Generated lambda functions will be serialized to file, but pretty prints (SymPy objects) can only exist in the System instance on which prepare is called.

**Notes**

Option `incremental` compares the md5 checksum of all var and service strings, and only regenerate for updated models.

**Examples**

If one needs to print out LaTeX-formatted equations in a Jupyter Notebook, one need to generate such equations with

```
import andes
sys = andes.prepare()
```

Alternatively, one can explicitly create a System and generate the code

```
import andes
sys = andes.System()
sys.prepare()
```

`andes.system.System.undill(self, autogen_stale=True)`

Reload generated function functions, from either the `$HOME/.andes/pycode` folder.

If no change is made to models, future calls to `prepare()` can be replaced with `undill()` for acceleration.

**Parameters****autogen\_stale: bool**

True to automatically call code generation if stale code is detected. Regardless of this option, codegen is trigger if importing existing code fails.

### 3.1.2 DAE Storage

`System.dae` is an instance of the numerical 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 algebraic 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**, , , and , 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.

### 3.1.3 Model and DAE Values

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 to each model.

The collection of values from models needs to follow protocols to avoid conflicts. Details are given in the subsection Variables.

`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

## Matrix Sparsity Patterns

The largest overhead in building and solving nonlinear equations is the building of Jacobian matrices. This is especially relevant when we use the implicit integration approach which algebraized 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. The first one is to use simple in-place add on sparse matrices, such as doing

```
self.fx += spmatrix(v, i, j, (n, n), 'd')
```

Although the implementation is simple, it involves creating and discarding temporary objects on the right hand side and, even worse, changing the sparse pattern of `self.fx`.

The second approach is to store the rows, columns and values in an array-like object and construct the Jacobians at the end. This approach is very efficient but has one caveat: it does not allow accessing the sparse matrix while building.

ANDES uses a pre-allocation approach to avoid the change of sparse patterns by filling values into a known the sparse matrix pattern matrix. 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 are only modifying existing values, it does not change the pattern and thus avoids memory copying. In addition, updating sparse matrices can be done with the exact same code as the first approach.

Still, this approach creates and discards temporary objects. It is however feasible to write a C function which takes three array-likes and modify the sparse matrices in place. This is feature to be developed, and our prototype shows a promising acceleration up to 50%.

`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.

### 3.1.4 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.

`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.

`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.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.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.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).

### 3.1.5 Configuration

System, models and routines have a member attribute `config` for model-specific or routine-specific configurations. System manages all configs, including saving to a config file and loading back.

`andes.system.System.save_config(self, file_path=None, overwrite=False)`

Save all system, model, and routine configurations to an rc-formatted file.

#### Parameters

##### `file_path`

[str, optional] path to the configuration file default to `~/andes/andes.rc`.

##### `overwrite`

[bool, optional] If file exists, True to overwrite without confirmation. Otherwise prompt for confirmation.

**Warning:** Saved config is loaded back and populated at system instance creation time. Configs from the config file takes precedence over default config values.

**Warning:** It is important to note that configs from files is passed to *model constructors* during instantiation. If one needs to modify config for a run, it needs to be done before instantiating System, or before running andes from command line. Directly modifying Model.config may not take effect or have side effect as for the current implementation.

## 3.2 Group

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 develop new models. The common variables and parameters are typically the interface when connecting different group models.

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 model in the same group could carry out similar function.

---

*GroupBase()*

Base class for groups.

---

### 3.2.1 andes.models.group.GroupBase

**class** andes.models.group.**GroupBase**

Base class for groups.

**\_\_init\_\_()**

## Methods

<code>add(idx, model)</code>	Register an idx from model_name to the group
<code>add_model(name, instance)</code>	Add a Model instance to group.
<code>doc([export])</code>	Return the documentation of the group in a string.
<code>doc_all([export])</code>	Return documentation of the group and its models.
<code>find_idx(keys, values[, allow_none, default])</code>	Find indices of devices that satisfy the given <code>key=value</code> condition.
<code>get(src, idx[, attr, allow_none, default])</code>	Based on the indexer, get the <code>attr</code> field of the <code>src</code> parameter or variable.
<code>get_field(src, idx, field)</code>	Helper function for retrieving an attribute of a member variable shared by models in this group.
<code>get_next_idx([idx, model_name])</code>	Get a no-conflict idx for a new device.
<code>idx2model(idx[, allow_none])</code>	Find model name for the given idx.
<code>idx2uid(idx)</code>	Convert idx to the 0-indexed unique index.
<code>set(src, idx, attr, value)</code>	Set the value of an attribute of a group property.
<code>set_backref(name, from_idx, to_idx)</code>	Set idxes to BackRef, and set them to models.

### GroupBase.add

`GroupBase.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

### GroupBase.add\_model

`GroupBase.add_model(name: str, instance)`

Add a Model instance to group.

#### Parameters

**name**

[str] Model name

**instance**

[Model] Model instance

#### Returns

**None**

### **GroupBase.doc**

`GroupBase.doc (export='plain')`

Return the documentation of the group in a string.

### **GroupBase.doc\_all**

`GroupBase.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**

### **GroupBase.find\_idx**

`GroupBase.find_idx (keys, values, allow_none=False, default=None)`

Find indices of devices that satisfy the given *key=value* condition.

This method iterates over all models in this group.

### **GroupBase.get**

`GroupBase.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.

**GroupBase.get\_field**

GroupBase.**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)`.

**GroupBase.get\_next\_idx**

GroupBase.**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.

**GroupBase.idx2model**

GroupBase.**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.

## GroupBase.idx2uid

GroupBase.**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

## GroupBase.set

GroupBase.**set**(*src*: str, *idx*, *attr*, *value*)

Set the value of an attribute of a group property. Performs self.<*src*>.<*attr*>[*idx*] = *value*.

The user needs to ensure that the property is shared by all models in this group.

### Parameters

#### src

[str] Name of property.

#### idx

[str, int, float, array-like] Indices of devices.

#### attr

[str, optional, default='v'] The internal attribute of the property to get. v for values, a for address, and e for equation value.

#### value

[array-like] New values to be set

### Returns

#### bool

True when successful.

## GroupBase.set\_backref

`GroupBase.set_backref(name, from_idx, to_idx)`

Set idxes to BackRef, and set them to models.

## Attributes

<code>class_name</code>	
<code>n</code>	Total number of devices.

## GroupBase.class\_name

`property GroupBase.class_name`

## GroupBase.n

`property GroupBase.n`

Total number of devices.

## 3.3 Models

This section introduces the modeling of power system devices. The terminology "model" is used to describe the mathematical representation of a *type* of device, such as synchronous generators or turbine governors. The terminology "device" is used to describe a particular instance of a model, for example, a specific generator.

To define a model in ANDES, two classes, `ModelData` and `Model` need to be utilized. Class `ModelData` is used for defining parameters that will be provided from input files. It provides API for adding data from devices and managing the data. Class `Model` is used for defining other non-input parameters, service variables, and DAE variables. It provides API for converting symbolic equations, storing Jacobian patterns, and updating equations.

The following classes are related to models:

<code>ModelData(*args[, three_params])</code>	Class for holding parameter data for a model.
<code>Model([system, config])</code>	Base class for power system DAE models.
<code>ModelCache()</code>	Class for caching the return value of callback functions.
<code>ModelCall()</code>	Class for storing generated function calls, Jacobian calls, and arguments.

### 3.3.1 andes.core.model.ModelData

```
class andes.core.model.ModelData(*args, three_params=True, **kwargs)
```

Class for holding parameter data for a model.

This class is designed to hold the parameter data separately from model equations. Models should inherit this class to define the parameters from input files.

Inherit this class to create the specific class for holding input parameters for a new model. The recommended name for the derived class is the model name with Data. For example, data for *GENROU* should be named *GENROUData*.

Parameters should be defined in the `__init__` function of the derived class.

Refer to `andes.core.param` for available parameter types.

#### Notes

Three default parameters are pre-defined in `ModelData` and will be inherited by all models. They are

- `idx`, unique device idx of type `andes.core.param.DataParam`
- `u`, connection status of type `andes.core.param.NumParam`
- `name`, (device name of type `andes.core.param.DataParam`)

In rare cases one does not want to define these three parameters, one can pass `three_params=True` to the constructor of `ModelData`.

#### Examples

If we want to build a class `PQData` (for static PQ load) with three parameters,  $V_n$ ,  $p_0$  and  $q_0$ , we can use the following

```
from andes.core.model import ModelData, Model
from andes.core.param import IdxParam, NumParam

class PQData(ModelData):
    super().__init__()
    self.Vn = NumParam(default=110,
                        info="AC voltage rating",
                        unit='kV', non_zero=True,
                        tex_name=r'V_n')
    self.p0 = NumParam(default=0,
                        info='active power load in system base',
                        tex_name=r'p_0', unit='p.u.')
    self.q0 = NumParam(default=0,
                        info='reactive power load in system base',
                        tex_name=r'q_0', unit='p.u.')
```

In this example, all the three parameters are defined as `andes.core.param.NumParam`. In the full `PQData` class, other types of parameters also exist. For example, to store the idx of `owner`, `PQData` uses

```
self.owner = IdxParam(model='Owner', info="owner idx")
```

## Attributes

### cache

A cache instance for different views of the internal data.

### flags

[dict] Flags to control the routine and functions that get called. If the model is using user-defined numerical calls, set `f_num`, `g_num` and `j_num` properly.

```
__init__(*args, three_params=True, **kwargs)
```

## Methods

<code>add(**kwargs)</code>	Add a device (an instance) to this model.
<code>as_df([vin])</code>	Export all parameters as a <code>pandas.DataFrame</code> object.
<code>as_df_local()</code>	Export local variable values and services to a DataFrame.
<code>as_dict([vin])</code>	Export all parameters as a dict.
<code>find_idx(keys, values[, allow_none, default])</code>	Find <code>idx</code> of devices whose values match the given pattern.
<code>find_param(prop)</code>	Find params with the given property and return in an <code>OrderedDict</code> .
<code>update_from_df(df[, vin])</code>	Update parameter values from a DataFrame.

## ModelData.add

```
ModelData.add(**kwargs)
```

Add a device (an instance) to this model.

### Parameters

#### kwargs

model parameters are collected into the kwargs dictionary

**Warning:** This function is not intended to be used directly. Use the `add` method from System so that the index can be registered correctly.

## ModelData.as\_df

ModelData.**as\_df** (*vin=False*)

Export all parameters as a *pandas.DataFrame* object. This function utilizes *as\_dict* for preparing data.

### Returns

#### DataFrame

A dataframe containing all model data. An *uid* column is added.

#### vin

[bool] If True, export all parameters from original input (*vin*).

## ModelData.as\_df\_local

ModelData.**as\_df\_local** ()

Export local variable values and services to a DataFrame.

## ModelData.as\_dict

ModelData.**as\_dict** (*vin=False*)

Export all parameters as a dict.

### Returns

#### dict

a dict with the keys being the *ModelData* parameter names and the values being an array-like of data in the order of adding. An additional *uid* key is added with the value default to range(n).

## ModelData.find\_idx

ModelData.**find\_idx** (*keys, values, allow\_none=False, default=False*)

Find *idx* of devices whose values match the given pattern.

### 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 corresponds to the key.

#### allow\_none

[bool, Sized] Allow key, value to be not found. Used by groups.

**default**  
 [bool] Default idx to return if not found (missing)

**Returns**

**list**  
 indices of devices

**ModelData.find\_param**

`ModelData.find_param(prop)`

Find params with the given property and return in an `OrderedDict`.

**Parameters**

**prop**  
 [str] Property name

**Returns**

`OrderedDict`

**ModelData.update\_from\_df**

`ModelData.update_from_df(df, vin=False)`

Update parameter values from a `DataFrame`.

Adding devices are not allowed.

### 3.3.2 andes.core.model.Model

**class** `andes.core.model.Model(system=None, config=None)`

Base class for power system DAE models.

After subclassing `ModelData`, subclass `Model`` to complete a DAE model. Subclasses of `Model` define DAE variables, services, and other types of parameters, in the constructor `__init__`.

### Examples

Take the static `PQ` as an example, the subclass of `Model`, `PQ`, should look like

```
class PQ(PQData, Model):
    def __init__(self, system, config):
        PQData.__init__(self) Model.__init__(self, system, config)
```

Since `PQ` is calling the base class constructors, it is meant to be the final class and not further derived. It inherits from `PQData` and `Model` and must call constructors in the order of `PQData` and `Model`. If the derived class of `Model` needs to be further derived, it should only derive from `Model` and use a name ending with `Base`. See `andes.models.synchronous.GENBase`.

Next, in *PQ*.*\_\_init\_\_*, set proper flags to indicate the routines in which the model will be used

```
self.flags.update({'pflow': True})
```

Currently, flags *pflow* and *tds* are supported. Both are *False* by default, meaning the model is neither used in power flow nor in time-domain simulation. **A very common pitfall is forgetting to set the flag.**

Next, the group name can be provided. A group is a collection of models with common parameters and variables. Devices' idx of all models in the same group must be unique. To provide a group name, use

```
self.group = 'StaticLoad'
```

The group name must be an existing class name in `andes.models.group`. The model will be added to the specified group and subject to the variable and parameter policy of the group. If not provided with a group class name, the model will be placed in the *Undefined* group.

Next, additional configuration flags can be added. Configuration flags for models are load-time variables, specifying the behavior of a model. They can be exported to an *andes.rc* file and automatically loaded when creating the *System*. Configuration flags can be used in equation strings, as long as they are numerical values. To add config flags, use

```
self.config.add(OrderedDict((('pq2z', 1), )))
```

It is recommended to use *OrderedDict* instead of *dict*, although the syntax is verbose. Note that booleans should be provided as integers (1 or 0), since *True* or *False* is interpreted as a string when loaded from the *rc* file and will cause an error.

Next, it's time for variables and equations! The *PQ* class does not have internal variables itself. It uses its *bus* parameter to fetch the corresponding *a* and *v* variables of buses. Equation wise, it imposes an active power and a reactive power load equation.

To define external variables from *Bus*, use

```
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')
```

Refer to the subsection Variables for more details.

The simplest *PQ* model will impose constant P and Q, coded as

```
self.a.e_str = "u * p"
self.v.e_str = "u * q"
```

where the *e\_str* attribute is the equation string attribute. *u* is the connectivity status. Any parameter, config, service or variable can be used in equation strings.

Three additional scalars can be used in equations: - *dae\_t* for the current simulation time (can be used if the model has flag *tds*). - *sys\_f* for system frequency (from `system.config.freq`). - *sys\_mva* for system base mva (from `system.config.mva`).

The above example is overly simplified. Our *PQ* model wants a feature to switch itself to a constant impedance if the voltage is out of the range (*vmin*, *vmax*). To implement this, we need to introduce

a discrete component called *Limiter*, which yields three arrays of binary flags, *zi*, *zl*, and *zu* indicating in-range, below lower-limit, and above upper-limit, respectively.

First, create an attribute *vcmp* as a *Limiter* instance

```
self.vcmp = Limiter(u=self.v, lower=self.vmin, upper=self.vmax,
                     enable=self.config.pq2z)
```

where *self.config.pq2z* is a flag to turn this feature on or off. After this line, we can use *vcmp\_zi*, *vcmp\_zl*, and *vcmp\_zu* in other equation strings.

```
self.a.e_str = "u * (p0 * vcmp_zi + " \
                "p0 * vcmp_zl * (v ** 2 / vmin ** 2) + " \
                "p0 * vcmp_zu * (v ** 2 / vmax ** 2))"

self.v.e_str = "u * (q0 * vcmp_zi + " \
                "q0 * vcmp_zl * (v ** 2 / vmin ** 2) + " \
                "q0 * vcmp_zu * (v ** 2 / vmax ** 2))"
```

Note that *PQ.a.e\_str* can use the three variables from *vcmp* even before defining *PQ.vcmp*, as long as *PQ.vcmp* is defined, because *vcmp\_zi* is just a string literal in *e\_str*.

The two equations above implement a piece-wise power injection equation. It selects the original power demand if within range, and uses the calculated power when out of range.

Finally, to let ANDES pick up the model, the model name needs to be added to *models/\_init\_\_.py*. Follow the examples in the *OrderedDict*, where the key is the file name, and the value is the class name.

## Attributes

### `num_params`

[OrderedDict] {name: instance} of numerical parameters, including internal and external ones

### `__init__(system=None, config=None)`

## Methods

<code>a_reset()</code>	Reset addresses to empty and reset flags.address to False.
<code>alter(src, idx, value)</code>	Alter values of input parameters or constant service.
<code>doc([max_width, export])</code>	Retrieve model documentation as a string.
<code>e_clear()</code>	Clear equation value arrays associated with all internal variables.
<code>externalize()</code>	Externalize internal data as a snapshot.
<code>f_numeric(**kwargs)</code>	Custom fcall functions.
<code>f_update()</code>	Evaluate differential equations.
<code>g_numeric(**kwargs)</code>	Custom gcall functions.

continues on next page

Table 1 – continued from previous page

<code>g_update()</code>	Evaluate algebraic equations.
<code>get(src, idx[, attr, allow_none, default])</code>	Get the value of an attribute of a model property.
<code>get_init_order()</code>	Get variable initialization order and send to <code>logger.info</code> .
<code>get_inputs([refresh])</code>	Get an OrderedDict of the inputs to the numerical function calls.
<code>get_md5()</code>	Return the md5 hash of concatenated equation strings.
<code>get_times()</code>	Get event switch_times from <code>TimerParam</code> .
<code>idx2uid(idx)</code>	Convert idx to the 0-indexed unique index.
<code>init(routine)</code>	Numerical initialization of a model.
<code>internalize()</code>	Internalize snapshot data.
<code>j_numeric(**kwargs)</code>	Custom numeric update functions.
<code>j_update()</code>	Update Jacobian elements.
<code>l_check_eq([init, niter])</code>	Call the <code>check_eq</code> method of discrete components to update equation-dependent flags.
<code>l_update_var(dae_t, *args[, niter, err])</code>	Call the <code>check_var</code> method of discrete components to update the internal status flags.
<code>list2array()</code>	Convert all the value attributes <code>v</code> to NumPy arrays.
<code>mock_refresh_inputs()</code>	Use mock data to fill the inputs.
<code>numba_jitify([parallel, cache, nopython])</code>	Convert equation residual calls, Jacobian calls, and variable service calls into JIT compiled functions.
<code>post_init_check()</code>	Post init checking.
<code>precompile()</code>	Trigger numba compilation for this model.
<code>prepare([quick, pycode_path, yapf_pycode])</code>	Symbolic processing and code generation.
<code>refresh_inputs()</code>	This is the helper function to refresh inputs.
<code>refresh_inputs_arg()</code>	Refresh inputs for each function with individual argument list.
<code>register_debug_equation(var_name)</code>	Helper function to register a variable for debugging the initialization.
<code>s_numeric(**kwargs)</code>	Custom service value functions.
<code>s_numeric_var(**kwargs)</code>	Custom variable service value functions.
<code>s_update()</code>	Update service equation values.
<code>s_update_post()</code>	Update post-initialization services.
<code>s_update_var()</code>	Update values of <code>andes.core.service.VarService</code> .
<code>set(src, idx, attr, value)</code>	Set the value of an attribute of a model property.
<code>set_backref(name, from_idx, to_idx)</code>	Helper function for setting idx-es to BackRef.
<code>set_in_use()</code>	Set the <code>in_use</code> attribute.
<code>solve_iter(name, kwargs)</code>	Solve iterative initialization.
<code>solve_iter_single(name, inputs, pos)</code>	Solve iterative initialization for one given device.
<code>store_sparse_pattern()</code>	Store rows and columns of the non-zeros in the Jacobians for building the sparsity pattern.

continues on next page

Table 1 – continued from previous page

<code>switch_action(dae_t)</code>	Call the switch actions.
<code>v_numeric(**kwargs)</code>	Custom variable initialization function.

### Model.a\_reset

`Model.a_reset()`

Reset addresses to empty and reset flags.address to `False`.

### Model.alter

`Model.alter(src, idx, value)`

Alter values of input parameters or constant service.

If the method operates on an input parameter, the new data should be in the same base as that in the input file. This function will convert `value` to per unit in the system base whenever necessary.

The values for storing the input data, i.e., the parameter's `vin` field, will be overwritten. As a result, altered values will be reflected in the dumped case file.

#### Parameters

##### `src`

[str] The parameter name to alter

##### `idx`

[str, float, int] The device to alter

##### `value`

[float] The desired value

### Model.doc

`Model.doc(max_width=78, export='plain')`

Retrieve model documentation as a string.

### Model.e\_clear

`Model.e_clear()`

Clear equation value arrays associated with all internal variables.

## Model.externalize

```
Model.externalize()
```

Externalize internal data as a snapshot.

## Model.f\_numeric

```
Model.f_numeric(**kwargs)
```

Custom fcall functions. Modify equations directly.

## Model.f\_update

```
Model.f_update()
```

Evaluate differential equations.

## Notes

In-place equations: added to the corresponding DAE array. Non-in-place equations: in-place set to internal array to overwrite old values (and avoid clearing).

## Model.g\_numeric

```
Model.g_numeric(**kwargs)
```

Custom gcall functions. Modify equations directly.

## Model.g\_update

```
Model.g_update()
```

Evaluate algebraic equations.

## Model.get

```
Model.get(src: str, idx, attr: str = 'v', allow_none=False, default=0.0)
```

Get the value of an attribute of a model property.

The return value is `self.<src>.<attr>[idx]`

### Parameters

#### src

[str] Name of the model property

#### idx

[str, int, float, array-like] Indices of the devices

**attr**

[str, optional, default='v'] The attribute of the property to get. v for values, a for address, and e for equation value.

**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****array-like**

`self.<src>.<attr>[idx]`

## Model.get\_init\_order

`Model.get_init_order()`

Get variable initialization order and send to *logger.info*.

## Model.get\_inputs

`Model.get_inputs(refresh=False)`

Get an OrderedDict of the inputs to the numerical function calls.

**Parameters****refresh**

[bool] Refresh the values in the dictionary. This is only used when the memory addresses of arrays change. After initialization, all array assignments are in place. To avoid overhead, refresh should not be used after initialization.

**Returns****OrderedDict**

The input name and value array pairs in an OrderedDict

## Notes

*dae.t* is now a numpy.ndarray which has stable memory. There is no need to refresh *dat\_t* in this version.

## Model.get\_md5

`Model.get_md5()`

Return the md5 hash of concatenated equation strings.

## Model.get\_times

`Model.get_times()`

Get event switch\_times from *TimerParam*.

### Returns

#### list

A list containing all switching times defined in TimerParams

## Model.idx2uid

`Model.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

## Model.init

`Model.init(routine)`

Numerical initialization of a model.

Initialization sequence: 1. Sequential initialization based on the order of definition 2. Use Newton-Krylov method for iterative initialization 3. Custom init

## Model.internalize

`Model.internalize()`

Internalize snapshot data.

## **Model.j\_numeric**

`Model.j_numeric(**kwargs)`

Custom numeric update functions.

This function should append indices to `_ifx`, `_jfx`, and append anonymous functions to `_vfx`. It is only called once by `store_sparse_pattern`.

## **Model.j\_update**

`Model.j_update()`

Update Jacobian elements.

Values are stored to `Model.triplets[jname]`, where `jname` is a jacobian name.

### **Returns**

**None**

## **Model.l\_check\_eq**

`Model.l_check_eq(init=False, niter=0, **kwargs)`

Call the `check_eq` method of discrete components to update equation-dependent flags.

This function should be called after equation updates. AntiWindup limiters use it to append pegged states to the `x_set` list.

### **Returns**

**None**

## **Model.l\_update\_var**

`Model.l_update_var(dae_t, *args, niter=None, err=None, **kwargs)`

Call the `check_var` method of discrete components to update the internal status flags.

The function is variable-dependent and should be called before updating equations.

### **Returns**

**None**

## Model.list2array

```
Model.list2array()
```

Convert all the value attributes  $v$  to NumPy arrays.

Value attribute arrays should remain in the same address afterwards. Namely, all assignments to value array should be operated in place (e.g., with  $[:]$ ).

## Model.mock\_refresh\_inputs

```
Model.mock_refresh_inputs()
```

Use mock data to fill the inputs.

This function is used to generate input data of the desired type to trigger JIT compilation.

## Model.numba jitify

```
Model.numba jitify(parallel=False, cache=True, nopython=True)
```

Convert equation residual calls, Jacobian calls, and variable service calls into JIT compiled functions.

This function can be enabled by setting `System.config.numba = 1`.

## Model.post\_init\_check

```
Model.post_init_check()
```

Post init checking. Warns if values of `InitChecker` are not True.

## Model.precompile

```
Model.precompile()
```

Trigger numba compilation for this model.

This function requires the system to be setup, i.e., memory allocated for storage.

## Model.prepare

```
Model.prepare(quick=False, pycode_path=None, yapf_pycode=False)
```

Symbolic processing and code generation.

## Model.refresh\_inputs

```
Model.refresh_inputs()
```

This is the helper function to refresh inputs.

The function collects object references into `OrderedDict` `self._input` and `self._input_z`.

### Returns

`None`

## Model.refresh\_inputs\_arg

```
Model.refresh_inputs_arg()
```

Refresh inputs for each function with individual argument list.

## Model.register\_debug\_equation

```
Model.register_debug_equation(var_name)
```

Helper function to register a variable for debugging the initialization.

This function needs to be called before calling `TDS.init()`, and logging level needs to be set to `DEBUG`.

## Model.s\_numeric

```
Model.s_numeric(**kwargs)
```

Custom service value functions. Modify `Service.v` directly.

## Model.s\_numeric\_var

```
Model.s_numeric_var(**kwargs)
```

Custom variable service value functions. Modify `VarService.v` directly.

This custom numerical function is evaluated at each step/iteration before equation update.

## Model.s\_update

```
Model.s_update()
```

Update service equation values.

This function is only evaluated at initialization. Service values are updated sequentially. The `v` attribute of services will be assigned at a new memory address.

## Model.s\_update\_post

Model.s\_update\_post()

Update post-initialization services.

## Model.s\_update\_var

Model.s\_update\_var()

Update values of andes.core.service.VarService.

## Model.set

Model.set(src, idx, attr, value)

Set the value of an attribute of a model property.

Performs self.<src>.<attr>[idx] = value. This method will not modify the input values from the case file that have not been converted to the system base. As a result, changes applied by this method will not affect the dumped case file.

To alter parameters and reflect it in the case file, use [alter\(\)](#) instead.

### Parameters

#### src

[str] Name of the model property

#### idx

[str, int, float, array-like] Indices of the devices

#### attr

[str, optional, default='v'] The internal attribute of the property to get. v for values, a for address, and e for equation value.

#### value

[array-like] New values to be set

### Returns

#### bool

True when successful.

## Model.set\_backref

Model.set\_backref(name, from\_idx, to\_idx)

Helper function for setting idx-es to BackRef.

## Model.set\_in\_use

Model.set\_in\_use()

Set the *in\_use* attribute. Called at the end of System.collect\_ref.

This function is overloaded by models with *BackRef* to disable calls when no model is referencing. Models with no back references will have internal variable addresses assigned but external addresses being empty.

For internal equations that have external variables, the row indices will be non-zeros, while the col indices will be empty, which causes an error when updating Jacobians.

Setting *self.in\_use* to False when  $\text{len}(\text{back\_ref\_instance.v}) == 0$  avoids this error. See COI.

## Model.solve\_iter

Model.solve\_iter(name, kwargs)

Solve iterative initialization.

## Model.solve\_iter\_single

Model.solve\_iter\_single(name, inputs, pos)

Solve iterative initialization for one given device.

## Model.store\_sparse\_pattern

Model.store\_sparse\_pattern()

Store rows and columns of the non-zeros in the Jacobians for building the sparsity pattern.

This function converts the internal 0-indexed equation/variable address to the numerical addresses for the loaded system.

Calling sequence: For each Jacobian name, *fx*, , *gx* and *gy*, store by a) generated constant and variable Jacobians c) user-provided constant and variable Jacobians, d) user-provided block constant and variable Jacobians

## Notes

If *self.n == 0*, skipping this function will avoid appending empty lists/arrays and non-empty values, which, as a combination, is not accepted by *kvxopt.spmatrix*.

## Model.switch\_action

`Model.switch_action(dae_t)`

Call the switch actions.

### Parameters

`dae_t`

[float] Current simulation time

### Returns

`None`

**Warning:** Timer exported from blocks are supposed to work but have not been tested.

## Model.v\_numeric

`Model.v_numeric(**kwargs)`

Custom variable initialization function.

### Attributes

`class_name`

Return the class name

## Model.class\_name

`property Model.class_name`

Return the class name

### 3.3.3 andes.core.model.ModelCache

`class andes.core.model.ModelCache`

Class for caching the return value of callback functions.

Check `ModelCache.__dict__.keys()` for fields.

`__init__()`

## Methods

<code>add_callback(name, callback)</code>	Add a cache attribute and a callback function for updating the attribute.
<code>refresh([name])</code>	Refresh the cached values

### ModelCache.add\_callback

`ModelCache.add_callback(name: str, callback)`

Add a cache attribute and a callback function for updating the attribute.

#### Parameters

##### **name**

[str] name of the cached function return value

##### **callback**

[callable] callback function for updating the cached attribute

### ModelCache.refresh

`ModelCache.refresh(name=None)`

Refresh the cached values

#### Parameters

##### **name**

[str, list, optional] name or list of cached to refresh, by default None for refreshing all

### 3.3.4 andes.core.model.ModelCall

`class andes.core.model.ModelCall`

Class for storing generated function calls, Jacobian calls, and arguments.

`__init__()`

## Methods

```
append_ijv(j_full_name, ii, jj, vv)
```

```
clear_ijv()
```

```
zip_ijv(j_full_name)
```

Return a zipped iterator for the rows, cols and vals for the specified matrix name.

### ModelCall.append\_ijv

```
ModelCall.append_ijv(j_full_name, ii, jj, vv)
```

### ModelCall.clear\_ijv

```
ModelCall.clear_ijv()
```

### ModelCall.zip\_ijv

```
ModelCall.zip_ijv(j_full_name)
```

Return a zipped iterator for the rows, cols and vals for the specified matrix name.

## 3.3.5 Cache

*ModelData* uses a lightweight class `andes.core.model.ModelCache` for caching its data as a dictionary or a pandas DataFrame. Four attributes are defined in *ModelData.cache*:

- *dict*: all data in a dictionary with the parameter names as keys and *v* values as arrays.
- *dict\_in*: the same as *dict* except that the values are from *v\_in*, the original input.
- *df*: all data in a pandas DataFrame.
- *df\_in*: the same as *df* except that the values are from *v\_in*.

Other attributes can be added by registering with *cache.add\_callback*.

```
andes.core.model.ModelCache.add_callback(self, name: str, callback)
```

Add a cache attribute and a callback function for updating the attribute.

### Parameters

#### name

[str] name of the cached function return value

#### callback

[callable] callback function for updating the cached attribute

### 3.3.6 Define Voltage Ratings

If a model is connected to an AC Bus or a DC Node, namely, if bus, bus1, node or node1 exists as parameter, it must provide the corresponding parameter, Vn, Vn1, Vdcn or Vdcn1, for rated voltages.

Controllers not connected to Bus or Node will have its rated voltages omitted and thus Vb = Vn = 1, unless one uses `andes.core.param.ExtParam` to retrieve the bus/node values.

As a rule of thumb, controllers not directly connected to the network shall use system-base per unit for voltage and current parameters. Controllers (such as a turbine governor) may inherit rated power from controlled models and thus power parameters will be converted consistently.

### Define a DAE Model

```
class andes.core.model.Model(system=None, config=None)
```

Base class for power system DAE models.

After subclassing `ModelData`, subclass `Model`` to complete a DAE model. Subclasses of `Model` define DAE variables, services, and other types of parameters, in the constructor `__init__`.

### Examples

Take the static PQ as an example, the subclass of `Model`, `PQ`, should look like

```
class PQ(PQData, Model):
    def __init__(self, system, config):
        PQData.__init__(self)
        Model.__init__(self, system, config)
```

Since `PQ` is calling the base class constructors, it is meant to be the final class and not further derived. It inherits from `PQData` and `Model` and must call constructors in the order of `PQData` and `Model`. If the derived class of `Model` needs to be further derived, it should only derive from `Model` and use a name ending with `Base`. See `andes.models.synchronous.genbase.GENBase`.

Next, in `PQ.__init__`, set proper flags to indicate the routines in which the model will be used

```
self.flags.update({'pflow': True})
```

Currently, flags `pflow` and `tds` are supported. Both are `False` by default, meaning the model is neither used in power flow nor in time-domain simulation. **A very common pitfall is forgetting to set the flag.**

Next, the group name can be provided. A group is a collection of models with common parameters and variables. Devices' idx of all models in the same group must be unique. To provide a group name, use

```
self.group = 'StaticLoad'
```

The group name must be an existing class name in `andes.models.group`. The model will be added to the specified group and subject to the variable and parameter policy of the group. If not provided with a group class name, the model will be placed in the `Undefined` group.

Next, additional configuration flags can be added. Configuration flags for models are load-time variables, specifying the behavior of a model. They can be exported to an *andes.rc* file and automatically loaded when creating the *System*. Configuration flags can be used in equation strings, as long as they are numerical values. To add config flags, use

```
self.config.add(OrderedDict((('pq2z', 1), )))
```

It is recommended to use *OrderedDict* instead of *dict*, although the syntax is verbose. Note that booleans should be provided as integers (1 or 0), since *True* or *False* is interpreted as a string when loaded from the *rc* file and will cause an error.

Next, it's time for variables and equations! The *PQ* class does not have internal variables itself. It uses its *bus* parameter to fetch the corresponding *a* and *v* variables of buses. Equation wise, it imposes an active power and a reactive power load equation.

To define external variables from *Bus*, use

```
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')
```

Refer to the subsection Variables for more details.

The simplest *PQ* model will impose constant P and Q, coded as

```
self.a.e_str = "u * p"
self.v.e_str = "u * q"
```

where the *e\_str* attribute is the equation string attribute. *u* is the connectivity status. Any parameter, config, service or variable can be used in equation strings.

Three additional scalars can be used in equations: - *dae\_t* for the current simulation time (can be used if the model has flag *tds*). - *sys\_f* for system frequency (from *system.config.freq*). - *sys\_mva* for system base mva (from *system.config.mva*).

The above example is overly simplified. Our *PQ* model wants a feature to switch itself to a constant impedance if the voltage is out of the range (*vmin*, *vmax*). To implement this, we need to introduce a discrete component called *Limiter*, which yields three arrays of binary flags, *zi*, *zl*, and *zu* indicating in-range, below lower-limit, and above upper-limit, respectively.

First, create an attribute *vcmp* as a *Limiter* instance

```
self.vcmp = Limiter(u=self.v, lower=self.vmin, upper=self.vmax,
                     enable=self.config.pq2z)
```

where *self.config.pq2z* is a flag to turn this feature on or off. After this line, we can use *vcmp\_zi*, *vcmp\_zl*, and *vcmp\_zu* in other equation strings.

```
self.a.e_str = "u * (p0 * vcmp_zi + " \
               "p0 * vcmp_zl * (v ** 2 / vmin ** 2) + " \
               "p0 * vcmp_zu * (v ** 2 / vmax ** 2))"
```

(continues on next page)

(continued from previous page)

```
self.v.e_str = "u * (q0 * vcmp_zi + " \
    "q0 * vcmp_zl * (v ** 2 / vmin ** 2) + " \
    "q0 * vcmp_zu * (v ** 2 / vmax ** 2))"
```

Note that  $PQ.a.e\_str$  can use the three variables from  $vcmp$  even before defining  $PQ.vcmp$ , as long as  $PQ.vcmp$  is defined, because  $vcmp\_zi$  is just a string literal in  $e\_str$ .

The two equations above implement a piece-wise power injection equation. It selects the original power demand if within range, and uses the calculated power when out of range.

Finally, to let ANDES pick up the model, the model name needs to be added to `models/__init__.py`. Follow the examples in the *OrderedDict*, where the key is the file name, and the value is the class name.

### Attributes

#### num\_params

[OrderedDict] {name: instance} of numerical parameters, including internal and external ones

### Dynamicity Under the Hood

The magic for automatic creation of variables are all hidden in `andes.core.model.Model.__setattr__()`, and the code is incredible simple. It sets the name, `tex_name`, and owner model of the attribute instance and, more importantly, does the book keeping. In particular, when the attribute is a `andes.core.block.Block` subclass, `__setattr__` captures the exported instances, recursively, and prepends the block name to exported ones. All these convenience owe to the dynamic feature of Python.

During the code generation phase, the symbols are created by checking the book-keeping attributes, such as `states`, `algebs`, and attributes in `Model.cache`.

In the numerical evaluation phase, `Model` provides a method, `andes.core.model.get_inputs()`, to collect the variable value arrays in a dictionary, which can be effortlessly passed as arguments to numerical functions.

### 3.3.7 Commonly Used Attributes in Models

The following `Model` attributes are commonly used for debugging. If the attribute is an *OrderedDict*, the keys are attribute names in str, and corresponding values are the instances.

- `params` and `params_ext`, two *OrderedDict* for internal (both numerical and non-numerical) and external parameters, respectively.
- `num_params` for numerical parameters, both internal and external.
- `states` and `algebs`, two *OrderedDict* for state variables and algebraic variables, respectively.
- `states_ext` and `algebs_ext`, two *OrderedDict* for external states and algebraics.
- `discrete`, an *OrderedDict* for discrete components.

- `blocks`, an *OrderedDict* for blocks.
- `services`, an *OrderedDict* for services with `v_str`.
- `services_ext`, an *OrderedDict* for externally retrieved services.

### 3.3.8 Attributes in `Model.cache`

Attributes in `Model.cache` are additional book-keeping structures for variables, parameters and services. The following attributes are defined.

- `all_vars`: all the variables.
- `all_vars_names`, a list of all variable names.
- `all_params`, all parameters.
- `all_params_names`, a list of all parameter names.
- `algebs_and_ext`, an *OrderedDict* of internal and external algebraic variables.
- `states_and_ext`, an *OrderedDict* of internal and external differential variables.
- `services_and_ext`, an *OrderedDict* of internal and external service variables.
- `vars_int`, an *OrderedDict* of all internal variables, states and then algebs.
- `vars_ext`, an *OrderedDict* of all external variables, states and then algebs.

## Equation Generation

`Model.syms`, an instance of `SymProcessor`, handles the symbolic to numeric generation when called. The equation generation is a multi-step process with symbol preparation, equation generation, Jacobian generation, initializer generation, and pretty print generation.

**class andes.core.SymProcessor(parent)**

A helper class for symbolic processing and code generation.

### Parameters

#### **parent**

[Model] The `Model` instance to process

### Attributes

#### **xy**

[sympy.Matrix] variables pretty print in the order of State, ExtState, Algeb, ExtAlgeb

#### **f**

[sympy.Matrix] differential equations pretty print

#### **g**

[sympy.Matrix] algebraic equations pretty print

**df**

[sympy.SparseMatrix] df /d (xy) pretty print

**dg**

[sympy.SparseMatrix] dg /d (xy) pretty print

**inputs\_dict**

[OrderedDict] All possible symbols in equations, including variables, parameters, discrete flags, and config flags. It has the same variables as what `get_inputs()` returns.

**vars\_dict**

[OrderedDict] variable-only symbols, which are useful when getting the Jacobian matrices.

**generate\_equations()**

Generate equations.

The pretty-print equations in matrices can be accessed in `self.f_matrix` and `self.g_matrix`.

**generate\_init()**

Generate initialization equations.

**generate\_jacobians(diag\_eps=1e-08)**

Generate Jacobians and store to corresponding triplets.

The internal indices of equations and variables are stored, alongside the lambda functions.

For example, `dg/dy` is a sparse matrix whose elements are `(row, col, val)`, where `row` and `col` are the internal indices, and `val` is the numerical lambda function. They will be stored to

`row -> self.calls._ig` `col -> self.calls._jgy` `val -> self.calls._vgy`

**generate\_symbols()**

Generate symbols for symbolic equation generations.

This function should run before other generate equations.

**Attributes****inputs\_dict**

[OrderedDict] name-symbol pair of all parameters, variables and configs

**vars\_dict**

[OrderedDict] name-symbol pair of all variables, in the order of `(states_and_ext + algebs_and_ext)`

Next, function `generate_equation` converts each DAE equation set to one numerical function calls and store it in `Model.calls`. The attributes for differential equation set and algebraic equation set are `f` and `g`. Differently, service variables will be generated one by one and store in an `OrderedDict` in `Model.calls.s`.

## Jacobian Storage

### 3.3.9 Abstract Jacobian Storage

Using the `.jacobian` method on `sympy.Matrix`, the symbolic Jacobians can be easily obtained. The complexity lies in the storage of the Jacobian elements. Observed that the Jacobian equation generation happens before any system is loaded, thus only the variable indices in the variable array is available. For each non-zero item in each Jacobian matrix, ANDES stores the equation index, variable index, and the Jacobian value (either a constant number or a callable function returning an array).

Note that, again, a non-zero entry in a Jacobian matrix can be either a constant or an expression. For efficiency, constant numbers and lambdified callables are stored separately. Constant numbers, therefore, can be loaded into the sparse matrix pattern when a particular system is given.

**Warning:** Data structure for the Jacobian storage has changed. Pending documentation update. Please check `andes.core.common.JacTriplet` class for more details.

The triplets, the equation (row) index, variable (column) index, and values (constant numbers or callable) are stored in `Model` attributes with the name of `_i, j, v}{Jacobian Name}{c or None}`, where `{i, j, v}` is a single character for row, column or value, `{Jacobian Name}` is a two-character Jacobian name chosen from `fx, fy, gx, and gy`, and `{c or None}` is either character `c` or no character, indicating whether it corresponds to the constants or non-constants in the Jacobian.

For example, the triplets for the constants in Jacobian `gy` are stored in `_igyc, _jgyc, and _vgyc`.

In terms of the non-constant entries in Jacobians, the callable functions are stored in the corresponding `_v{Jacobian Name}` array. Note the differences between, for example, `_vgy` and `_vgyc`: `_vgy` is a list of callables, while `_vgyc` is a list of constant numbers.

### 3.3.10 Concrete Jacobian Storage

When a specific system is loaded and the addresses are assigned to variables, the abstract Jacobian triplets, more specifically, the rows and columns, are replaced with the array of addresses. The new addresses and values will be stored in `Model` attributes with the names `{i, j, v}{Jacobian Name}{c or None}`. Note that there is no underscore for the concrete Jacobian triplets.

For example, if model `PV` has a list of variables `[p, q, a, v]`. The equation associated with `p` is `- u * p0`, and the equation associated with `q` is `u * (v0 - v)`. Therefore, the derivative of equation `v0 - v` over `v` is `-u`. Note that `u` is unknown at generation time, thus the value is NOT a constant and should go `vgy`.

The values in `_igy, _jgy` and `_vgy` contains, respectively, 1, 3, and a lambda function which returns `-u`.

When a specific system is loaded, for example, a 5-bus system, the addresses for the `q` and `v` are `[11, 13, 15, and [5, 7, 9]]`. `PV.igy` and `PV.jgy` will thus query the corresponding address list based on `PV._igy` and `PV._jgy` and store `[11, 13, 15, and [5, 7, 9]]`.

## Initialization

Value providers such as services and DAE variables need to be initialized. Services are initialized before any DAE variable. Both Services and DAE Variables are initialized *sequentially* in the order of declaration.

Each Service, in addition to the standard `v_str` for symbolic initialization, provides a `v_numeric` hook for specifying a custom function for initialization. Custom initialization functions for DAE variables, are lumped in a single function in `Model.v_numeric`.

ANDES has an *experimental* Newton-Krylov method based iterative initialization. All DAE variables with `v_iter` will be initialized using the iterative approach

## Additional Numerical Equations

Additional numerical equations are allowed to complete the "hybrid symbolic-numeric" framework. Numerical function calls are useful when the model DAE is non-standard or hard to be generalized. Since the symbolic-to-numeric generation is an additional layer on top of the numerical simulation, it is fundamentally the same as user-provided numerical function calls.

ANDES provides the following hook functions in each `Model` subclass for custom numerical functions:

- `v_numeric`: custom initialization function
- `s_numeric`: custom service value function
- `g_numeric`: custom algebraic equations; update the `e` of the corresponding variable.
- `f_numeric`: custom differential equations; update the `e` of the corresponding variable.
- `j_numeric`: custom Jacobian equations; the function should append to `_i`, `_j` and `_v` structures.

For most models, numerical function calls are unnecessary and not recommended as it increases code complexity. However, when the data structure or the DAE are difficult to generalize in the symbolic framework, the numerical equations can be used.

For interested readers, see the `COI` symbolic implementation which calculated the center-of-inertia speed of generators. The `COI` could have been implemented numerically with for loops instead of `NumReduce`, `NumRepeat` and external variables.

## 3.4 Atomic Types

ANDES contains three types of atom classes for building DAE models. These types are parameter, variable and service.

### 3.4.1 Value Provider

Before addressing specific atom classes, the terminology *v-provider*, and *e-provider* are discussed. A value provider class (or *v-provider* for short) references any class with a member attribute named *v*, which should be a list or a 1-dimensional array of values. For example, all parameter classes are v-providers, since a parameter class should provide values for that parameter.

---

**Note:** In fact, all types of atom classes are v-providers, meaning that an instance of an atom class must contain values.

---

The values in the *v* attribute of a particular instance are values that will substitute the instance for computation. If in a model, one has a parameter

```
self.v0 = NumParam()
self.b = NumParam()

# where self.v0.v = np.array([1., 1.05, 1.1]
#   and self.b.v = np.array([10., 10., 10.])
```

Later, this parameter is used in an equation, such as

```
self.v = ExtAlgeb(model='Bus', src='v',
                   indexer=self.bus,
                   e_str='v0 **2 * b')
```

While computing  $v0^{**2} * b$ , *v0* and *b* will be substituted with the values in *self.v0.v* and *self.b.v*.

Sharing this interface *v* allows interoperability among parameters and variables and services. In the above example, if one defines *v0* as a *ConstService* instance, such as

```
self.v0 = ConstService(v_str='1.0')
```

Calculations will still work without modification.

### 3.4.2 Equation Provider

Similarly, an equation provider class (or *e-provider*) references any class with a member attribute named *e*, which should be a 1-dimensional array of values. The values in the *e* array are the results from the equation and will be summed to the numerical DAE at the addresses specified by the attribute *a*.

---

**Note:** Currently, only variables are *e-provider* types.

---

If a model has an external variable that links to Bus.v (voltage), such as

```
self.v = ExtAlgeb(model='Bus', src='v',
                   indexer=self.bus,
                   e_str='v0 **2 * b')
```

The addresses of the corresponding voltage variables will be retrieved into `self.v.a`, and the equation evaluation results will be stored in `self.v.e`

## 3.5 Parameters

### 3.5.1 Background

Parameter is a type of building atom for DAE models. Most parameters are read directly from an input file and passed to equation, and other parameters can be calculated from existing parameters.

The base class for parameters in ANDES is `BaseParam`, which defines interfaces for adding values and checking the number of values. `BaseParam` has its values stored in a plain list, the member attribute `v`. Subclasses such as `NumParam` stores values using a NumPy ndarray.

An overview of supported parameters is given below.

<code>BaseParam</code> ([default, name, tex_name, info, ...])	The base parameter class.
<code>DataParam</code> ([default, name, tex_name, info, ...])	An alias of the <code>BaseParam</code> class.
<code>IdxParam</code> ([default, name, tex_name, info, ...])	An alias of <code>BaseParam</code> with an additional storage of the owner model name
<code>NumParam</code> (default, name, tex_name, info, ...)	A computational numerical parameter.
<code>ExtParam</code> (model, src[, indexer, vtype, ...])	A parameter whose values are retrieved from an external model or group.
<code>TimerParam</code> ([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.

## 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.

## 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.

**\_\_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

<code>add([value])</code>	Add a new parameter value (from a new device of the owner model) to the v list.
<code>get_names()</code>	Return <code>self.name</code> in a list.
<code>get_property(property_name)</code>	Check the boolean value of the given property.
<code>set(pos, attr, value)</code>	Set attributes of the <code>BaseParam</code> class to new values at the given positions.
<code>set_all(attr, value)</code>	Set attributes of the <code>BaseParam</code> class to new values for all positions.

### BaseParam.add

`BaseParam.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.

### BaseParam.get\_names

`BaseParam.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

## BaseParam.get\_property

BaseParam.**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.

## BaseParam.set

BaseParam.**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

## BaseParam.set\_all

BaseParam.**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

## Attributes

<code>class_name</code>	Return the class name.
<code>n</code>	Return the count of elements in the value array.

### `BaseParam.class_name`

**property** `BaseParam.class_name`

Return the class name.

### `BaseParam.n`

**property** `BaseParam.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

<code>add([value])</code>	Add a new parameter value (from a new device of the owner model) to the v list.
<code>get_names()</code>	Return <code>self.name</code> in a list.
<code>get_property(property_name)</code>	Check the boolean value of the given property.
<code>set(pos, attr, value)</code>	Set attributes of the <code>BaseParam</code> class to new values at the given positions.
<code>set_all(attr, value)</code>	Set attributes of the <code>BaseParam</code> class to new values for all positions.

### DataParam.add

`DataParam.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.

### DataParam.get\_names

`DataParam.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

## DataParam.get\_property

DataParam.**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.

## DataParam.set

DataParam.**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

## DataParam.set\_all

DataParam.**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

## Attributes

<code>class_name</code>	Return the class name.
<code>n</code>	Return the count of elements in the value array.

### DataParam.class\_name

**property** DataParam.`class_name`

Return the class name.

### DataParam.n

**property** DataParam.`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)
```

An alias of *BaseParam* with an additional storage of the owner model name

This class is intended for storing *idx* into other models. It can be used in the future for data consistency check.

## Notes

This will be useful when, for example, one connects two TGs to one SynGen.

## Examples

A PQ model connected to Bus model will have the following code

```
class PQModel(...):
    def __init__(...):
        ...
        self.bus = IdxParam(model='Bus')
```

---

**`__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*)

## Methods

<code>add([value])</code>	Add a new parameter value (from a new device of the owner model) to the v list.
<code>get_names()</code>	Return <code>self.name</code> in a list.
<code>get_property(property_name)</code>	Check the boolean value of the given property.
<code>set(pos, attr, value)</code>	Set attributes of the <code>BaseParam</code> class to new values at the given positions.
<code>set_all(attr, value)</code>	Set attributes of the <code>BaseParam</code> class to new values for all positions.

### IdxParam.add

`IdxParam.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.

### IdxParam.get\_names

`IdxParam.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

## **IdxParam.get\_property**

`IdxParam.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.**

## **IdxParam.set**

`IdxParam.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

## **IdxParam.set\_all**

`IdxParam.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

## Attributes

<code>class_name</code>	Return the class name.
<code>n</code>	Return the count of elements in the value array.

### `IdxParam.class_name`

**property** `IdxParam.class_name`

Return the class name.

### `IdxParam.n`

**property** `IdxParam.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.

## 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, vrangle: ~typing.List | ~typing.Tuple | None = None, vtype: ~typing.Type | None = <class 'float'>, icanvert: ~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

<code>add([value])</code>	Add a value to the parameter value list.
<code>get_names()</code>	Return <code>self.name</code> in a list.
<code>get_property(property_name)</code>	Check the boolean value of the given property.
<code>restore()</code>	Restore parameter to the original input by copying <code>self.vin</code> to <code>self.v</code> .
<code>set(pos, attr, value)</code>	Set attributes of the BaseParam class to new values at the given positions.
<code>set_all(attr, value)</code>	Set attributes of the BaseParam class to new values for all positions.
<code>set_pu_coeff(coeff)</code>	Store p.u.
<code>to_array()</code>	Converts field <code>v</code> to the NumPy array type.

## NumParam.add

NumParam.**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.

**See also:**

*BaseParam.add*

the add method of BaseParam

## NumParam.get\_names

NumParam.**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

## NumParam.get\_property

NumParam.**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.**

## NumParam.restore

NumParam.**restore**()

Restore parameter to the original input by copying self.vin to self.v.

*pu\_coeff* will not be overwritten.

## NumParam.set

NumParam.**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

## NumParam.set\_all

NumParam.**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

## NumParam.set\_pu\_coeff

NumParam.**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

## NumParam.to\_array

`NumParam.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.

## Attributes

<code>class_name</code>	Return the class name.
<code>n</code>	Return the count of elements in the value array.

## NumParam.class\_name

**property** `NumParam.class_name`

Return the class name.

## NumParam.n

**property** `NumParam.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**

<code>add([value])</code>	ExtParam has an empty <code>add</code> method.
<code>get_names()</code>	Return <code>self.name</code> in a list.
<code>get_property(property_name)</code>	Check the boolean value of the given property.
<code>link_external(ext_model)</code>	Update parameter values provided by external models.
<code>restore()</code>	ExtParam has an empty <code>restore</code> method
<code>set(pos, attr, value)</code>	Set attributes of the <code>BaseParam</code> class to new values at the given positions.
<code>set_all(attr, value)</code>	Set attributes of the <code>BaseParam</code> class to new values for all positions.
<code>set_pu_coeff(coeff)</code>	Store p.u.
<code>to_array()</code>	Convert to array when <code>d_type</code> is not <code>str</code>

**ExtParam.add**

`ExtParam.add (value=None)`

ExtParam has an empty `add` method.

**ExtParam.get\_names**

`ExtParam.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

## ExtParam.get\_property

ExtParam.**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.

## ExtParam.link\_external

ExtParam.**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.

## ExtParam.restore

ExtParam.**restore** ()

ExtParam has an empty *restore* method

## ExtParam.set

ExtParam.**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

## **ExtParam.set\_all**

`ExtParam.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

## **ExtParam.set\_pu\_coeff**

`ExtParam.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

## **ExtParam.to\_array**

`ExtParam.to_array()`

Convert to array when d\_type is not str

## **Attributes**

<code>class_name</code>	Return the class name.
<code>n</code>	Return the count of elements in the value array.

## **ExtParam.class\_name**

**property** `ExtParam.class_name`

Return the class name.

## ExtParam.n

```
property ExtParam.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 toggling 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
```

```
__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

<code>add([value])</code>	Add a value to the parameter value list.
<code>get_names()</code>	Return <code>self.name</code> in a list.
<code>get_property(property_name)</code>	Check the boolean value of the given property.
<code>is_time(dae_t)</code>	Element-wise check if the DAE time is the same as the parameter value.
<code>restore()</code>	Restore parameter to the original input by copying <code>self.vin</code> to <code>self.v</code> .
<code>set(pos, attr, value)</code>	Set attributes of the <code>BaseParam</code> class to new values at the given positions.
<code>set_all(attr, value)</code>	Set attributes of the <code>BaseParam</code> class to new values for all positions.
<code>set_pu_coeff(coeff)</code>	Store p.u.
<code>to_array()</code>	Converts field <code>v</code> to the NumPy array type.

### TimerParam.add

`TimerParam.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.

**See also:**

`BaseParam.add`

the add method of `BaseParam`

### TimerParam.get\_names

`TimerParam.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

## TimerParam.get\_property

TimerParam.**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.

## TimerParam.is\_time

TimerParam.**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.

## TimerParam.restore

TimerParam.**restore** ()

Restore parameter to the original input by copying *self.vin* to *self.v*.

*pu\_coeff* will not be overwritten.

## TimerParam.set

TimerParam.**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

## TimerParam.set\_all

TimerParam.**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

## TimerParam.set\_pu\_coeff

TimerParam.**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

## TimerParam.to\_array

`TimerParam.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.

## Attributes

<code>class_name</code>	Return the class name.
<code>n</code>	Return the count of elements in the value array.

### TimerParam.class\_name

**property** `TimerParam.class_name`

Return the class name.

### TimerParam.n

**property** `TimerParam.n`

Return the count of elements in the value array.

## 3.6 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. The base class for variables is `BaseVar`. In this subsection, `BaseVar` is used to represent any subclass of `VarBase` list in the table below.

<code>BaseVar</code> ([name, tex_name, info, unit, v_str, ...])	Base variable class.
<code>ExtVar</code> (model, src[, indexer, allow_none, ...])	Algebraic variable that links to an external model.
<code>State</code> ([name, tex_name, info, unit, v_str, ...])	Differential variable class, an alias of the <code>BaseVar</code> .
<code>Algeb</code> ([name, tex_name, info, unit, v_str, ...])	Algebraic variable class, an alias of <code>andes.core.var.BaseVar</code> .
<code>ExtState</code> (model, src[, indexer, allow_none, ...])	External state variable type.
<code>ExtAlgeb</code> (model, src[, indexer, allow_none, ...])	External algebraic variable type.
<code>AliasState</code> (var, **kwargs)	Alias state variable.
<code>AliasAlgeb</code> (var, **kwargs)	Alias algebraic variable.

### 3.6.1 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 assign 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  $0 = v_{\text{iter}}$

**\_\_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.

### BaseVar.get\_names

`BaseVar.get_names()`

### BaseVar.reset

`BaseVar.reset()`

Reset the internal numpy arrays and flags.

### BaseVar.set\_address

`BaseVar.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

## BaseVar.set\_arrays

`BaseVar.set_arrays(dae, inplace=True, alloc=True)`

Set the equation and values arrays.

### Parameters

#### dae

[DAE] Reference to System.dae

## Attributes

`class_name`

## BaseVar.class\_name

`property BaseVar.class_name`

### 3.6.2 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()*

<i>link_external(ext_model)</i>	Update variable addresses provided by external models
<i>reset()</i>	Reset the internal numpy arrays and flags.
<i>set_address(addr[, contiguous])</i>	Assigns address for equation RHS.
<i>set_arrays(dae[, inplace, alloc])</i>	Access dae.h or dae.i for the RHS of external variables when e_str exists..

**ExtVar.get\_names**

```
ExtVar.get_names()
```

**ExtVar.link\_external**

```
ExtVar.link_external(ext_model)
```

Update variable addresses provided by external models

This method sets attributes including *parent\_model*, *parent\_instance*, *uid*, *a*, *n*, *e\_code* and *v\_code*. It initializes the *e* and *v* to zero.

**Parameters****ext\_model**

[Model] Instance of the parent model

**Returns**

**None**

**Warning:** *link\_external* does not check if the ExtVar type is the same as the original variable to reduce performance overhead. It will be a silent error (a dimension too small error from *dae.build\_pattern*) if a model uses *ExtAlgeb* to access a *State*, or vice versa.

**ExtVar.reset**

```
ExtVar.reset()
```

Reset the internal numpy arrays and flags.

**ExtVar.set\_address**

```
ExtVar.set_address(addr, contiguous=False)
```

Assigns address for equation RHS.

**ExtVar.set\_arrays**

```
ExtVar.set_arrays(dae, inplace=True, alloc=True)
```

Access *dae.h* or *dae.i* for the RHS of external variables when *e\_str* exists..

## Attributes

```
class_name
```

### ExtVar.class\_name

```
property ExtVar.class_name
```

## 3.6.3 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.

## 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`.

### Attributes

#### `e_code`

[str] Equation code string, equals string literal `f`

#### `v_code`

[str] Variable code string, equals string literal `x`

```
__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

<code>get_names()</code>	
<code>reset()</code>	Reset the internal numpy arrays and flags.
<code>set_address(addr[, contiguous])</code>	Set the address of internal variables.
<code>set_arrays(dae[, inplace, alloc])</code>	Set the equation and values arrays.

```
State.get_names()
```

### State.reset

```
State.reset()
```

Reset the internal numpy arrays and flags.

### State.set\_address

```
State.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

## State.set\_arrays

```
State.set_arrays(dae, inplace=True, alloc=True)
```

Set the equation and values arrays.

### Parameters

**dae**

[DAE] Reference to System.dae

## Attributes

```
class_name
```

```
e_code
```

```
v_code
```

## State.class\_name

```
property State.class_name
```

## State.e\_code

```
State.e_code = 'f'
```

## State.v\_code

```
State.v_code = 'x'
```

## 3.6.4 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.

## 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  $\frac{d(x + z)}{d(y)}$  is zero.

### Attributes

#### `e_code`

[str] Equation code string, equals string literal  $g$

#### `v_code`

[str] Variable code string, equals string literal  $y$

```
__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

---

<code>get_names()</code>	
<code>reset()</code>	Reset the internal numpy arrays and flags.
<code>set_address(addr[, contiguous])</code>	Set the address of internal variables.
<code>set_arrays(dae[, inplace, alloc])</code>	Set the equation and values arrays.

---

### `Algeb.get_names`

`Algeb.get_names()`

### `Algeb.reset`

`Algeb.reset()`

Reset the internal numpy arrays and flags.

## Algeb.set\_address

Algeb.**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

## Algeb.set\_arrays

Algeb.**set\_arrays** (*dae*, *inplace=True*, *alloc=True*)

Set the equation and values arrays.

### Parameters

#### dae

[DAE] Reference to System.dae

## Attributes

---

<i>class_name</i>
<i>e_code</i>
<i>v_code</i>

---

## Algeb.class\_name

**property** Algeb.**class\_name**

## Algeb.e\_code

Algeb.**e\_code** = 'g'

**Algeb.v\_code**

```
Algeb.v_code = 'y'
```

**3.6.5 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()`

<code>link_external(ext_model)</code>	Update variable addresses provided by external models
<code>reset()</code>	Reset the internal numpy arrays and flags.
<code>set_address(addr[, contiguous])</code>	Assigns address for equation RHS.
<code>set_arrays(dae[, inplace, alloc])</code>	Access dae.h or dae.i for the RHS of external variables when e_str exists..

---

## ExtState.get\_names

```
ExtState.get_names()
```

## ExtState.link\_external

```
ExtState.link_external(ext_model)
```

Update variable addresses provided by external models

This method sets attributes including *parent\_model*, *parent\_instance*, *uid*, *a*, *n*, *e\_code* and *v\_code*. It initializes the *e* and *v* to zero.

### Parameters

#### ext\_model

[Model] Instance of the parent model

### Returns

None

**Warning:** *link\_external* does not check if the ExtVar type is the same as the original variable to reduce performance overhead. It will be a silent error (a dimension too small error from *dae.build\_pattern*) if a model uses *ExtAlgeb* to access a *State*, or vice versa.

## ExtState.reset

```
ExtState.reset()
```

Reset the internal numpy arrays and flags.

## ExtState.set\_address

```
ExtState.set_address(addr, contiguous=False)
```

Assigns address for equation RHS.

## ExtState.set\_arrays

```
ExtState.set_arrays(dae, inplace=True, alloc=True)
```

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
```

---

### **ExtState.class\_name**

```
property ExtState.class_name
```

### **ExtState.e\_code**

```
ExtState.e_code = 'f'
```

### **ExtState.r\_code**

```
ExtState.r_code = 'h'
```

### **ExtState.t\_const**

```
ExtState.t_const = None
```

### **ExtState.v\_code**

```
ExtState.v_code = 'x'
```

### 3.6.6 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()
```

<code>link_external(ext_model)</code>	Update variable addresses provided by external models
<code>reset()</code>	Reset the internal numpy arrays and flags.
<code>set_address(addr[, contiguous])</code>	Assigns address for equation RHS.
<code>set_arrays(dae[, inplace, alloc])</code>	Access dae.h or dae.i for the RHS of external variables when e_str exists..

---

## **ExtAlgeb.get\_names**

`ExtAlgeb.get_names()`

## **ExtAlgeb.link\_external**

`ExtAlgeb.link_external(ext_model)`

Update variable addresses provided by external models

This method sets attributes including `parent_model`, `parent_instance`, `uid`, `a`, `n`, `e_code` and `v_code`. It initializes the `e` and `v` to zero.

### **Parameters**

#### **ext\_model**

[Model] Instance of the parent model

### **Returns**

**None**

**Warning:** `link_external` does not check if the `ExtVar` type is the same as the original variable to reduce performance overhead. It will be a silent error (a dimension too small error from `dae.build_pattern`) if a model uses `ExtAlgeb` to access a `State`, or vice versa.

## **ExtAlgeb.reset**

`ExtAlgeb.reset()`

Reset the internal numpy arrays and flags.

## **ExtAlgeb.set\_address**

`ExtAlgeb.set_address(addr, contiguous=False)`

Assigns address for equation RHS.

## **ExtAlgeb.set\_arrays**

`ExtAlgeb.set_arrays(dae, inplace=True, alloc=True)`

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
```

---

### ExtAlgeb.class\_name

```
property ExtAlgeb.class_name
```

### ExtAlgeb.e\_code

```
ExtAlgeb.e_code = 'g'
```

### ExtAlgeb.r\_code

```
ExtAlgeb.r_code = 'i'
```

### ExtAlgeb.v\_code

```
ExtAlgeb.v_code = 'y'
```

## 3.6.7 andes.core.var.AliasState

```
class andes.core.var.AliasState(var, **kwargs)
```

Alias state variable.

Refer to the docs of AliasAlgeb.

```
__init__(var, **kwargs)
```

## Methods

<code>get_names()</code>	
<code>link_external(ext_model)</code>	Update variable addresses provided by external models
<code>reset()</code>	Reset the internal numpy arrays and flags.
<code>set_address(addr[, contiguous])</code>	Assigns address for equation RHS.
<code>set_arrays(dae[, inplace, alloc])</code>	Access <code>dae.h</code> or <code>dae.i</code> for the RHS of external variables when <code>e_str</code> exists..

### **AliasState.get\_names**

`AliasState.get_names()`

### **AliasState.link\_external**

`AliasState.link_external(ext_model)`

Update variable addresses provided by external models

This method sets attributes including `parent_model`, `parent_instance`, `uid`, `a`, `n`, `e_code` and `v_code`. It initializes the `e` and `v` to zero.

#### Parameters

##### **ext\_model**

[Model] Instance of the parent model

#### Returns

##### **None**

**Warning:** `link_external` does not check if the `ExtVar` type is the same as the original variable to reduce performance overhead. It will be a silent error (a dimension too small error from `dae.build_pattern`) if a model uses `ExtAlgeb` to access a `State`, or vice versa.

### **AliasState.reset**

`AliasState.reset()`

Reset the internal numpy arrays and flags.

### AliasState.set\_address

```
AliasState.set_address(addr, contiguous=False)
```

Assigns address for equation RHS.

### AliasState.set\_arrays

```
AliasState.set_arrays(dae, inplace=True, alloc=True)
```

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
```

### AliasState.class\_name

```
property AliasState.class_name
```

### AliasState.e\_code

```
AliasState.e_code = 'f'
```

### AliasState.r\_code

```
AliasState.r_code = 'h'
```

**AliasState.t\_const**

```
AliasState.t_const = None
```

**AliasState.v\_code**

```
AliasState.v_code = 'x'
```

**3.6.8 andes.core.var.AliasAlgeb**

```
class andes.core.var.AliasAlgeb(var, **kwargs)
```

Alias algebraic variable. Essentially ExtAlgeb that links to 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**

<code>get_names()</code>	
<code>link_external(ext_model)</code>	Update variable addresses provided by external models
<code>reset()</code>	Reset the internal numpy arrays and flags.
<code>set_address(addr[, contiguous])</code>	Assigns address for equation RHS.
<code>set_arrays(dae[, inplace, alloc])</code>	Access dae.h or dae.i for the RHS of external variables when e_str exists..

**AliasAlgeb.get\_names**

```
AliasAlgeb.get_names()
```

## AliasAlgeb.link\_external

AliasAlgeb.**link\_external** (*ext\_model*)

Update variable addresses provided by external models

This method sets attributes including *parent\_model*, *parent\_instance*, *uid*, *a*, *n*, *e\_code* and *v\_code*. It initializes the *e* and *v* to zero.

### Parameters

#### **ext\_model**

[Model] Instance of the parent model

### Returns

**None**

**Warning:** *link\_external* does not check if the ExtVar type is the same as the original variable to reduce performance overhead. It will be a silent error (a dimension too small error from *dae.build\_pattern*) if a model uses *ExtAlgeb* to access a *State*, or vice versa.

## AliasAlgeb.reset

AliasAlgeb.**reset** ()

Reset the internal numpy arrays and flags.

## AliasAlgeb.set\_address

AliasAlgeb.**set\_address** (*addr*, *contiguous=False*)

Assigns address for equation RHS.

## AliasAlgeb.set\_arrays

AliasAlgeb.**set\_arrays** (*dae*, *inplace=True*, *alloc=True*)

Access *dae.h* or *dae.i* for the RHS of external variables when *e\_str* exists..

## Attributes

<code>class_name</code>
<code>e_code</code>
<code>r_code</code>
<code>v_code</code>

### **AliasAlgeb.class\_name**

```
property AliasAlgeb.class_name
```

### **AliasAlgeb.e\_code**

```
AliasAlgeb.e_code = 'g'
```

### **AliasAlgeb.r\_code**

```
AliasAlgeb.r_code = 'i'
```

### **AliasAlgeb.v\_code**

```
AliasAlgeb.v_code = 'y'
```

Note that equations associated with state variables are in the form of  $M\dot{x} = f(x, y)$ , where  $x$  are the differential variables,  $y$  are the algebraic variables, and  $M$  is the mass matrix, and  $f$  are the right-hand side of differential equations. Equations associated with algebraic variables take the form of  $0 = g$ , where  $g$  are the equation right-hand side

*BaseVar* has two types: the differential variable type *State* and the algebraic variable type *Algeb*. State variables are described by differential equations, whereas algebraic variables are described by algebraic equations. State variables can only change continuously, while algebraic variables can be discontinuous.

Based on the model the variable is defined, variables can be internal or external. Most variables are internal and only appear in equations in the same model. Some models have "public" variables that can be accessed by other models. For example, a *Bus* defines  $v$  for the voltage magnitude. Each device attached to a particular bus needs to access the value and impose the reactive power injection. It can be done with *ExtAlgeb* or *ExtState*, which links with an existing variable from a model or a group.

### 3.6.9 Variable, Equation and Address

Subclasses of *BaseVar* are value providers and equation providers. Each *BaseVar* 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. In the process of 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 *BaseVar* also stores addresses of this variable, for all devices, in its member attribute *a*. The addresses are *0-based* indices into the numerical DAE array, *f* or *g*, based on the variable type.

For example, *Bus* has `self.a = Algeb()` as the voltage phase angle variable. For a 5-bus system, *Bus.a.a* stores the addresses of the *a* variable for all the five Bus devices. Conventionally, *Bus.a.a* will be assigned `np.array([0, 1, 2, 3, 4])`.

### 3.6.10 Value and Equation Strings

The most important feature of the symbolic framework is allowing to define equations using strings. There are three types of strings for a variable, stored in the following member attributes, respectively:

- *v\_str*: equation string for **explicit** initialization in the form of  $v = v_{str}(x, y)$ .
- *v\_iter*: equation string for **implicit** initialization in the form of  $v_{iter}(x, y) = 0$
- *e\_str*: equation string for (full or part of) the differential or algebraic equation.

The difference between *v\_str* and *v\_iter* should be clearly noted. *v\_str* evaluates directly into the initial value, while all *v\_iter* equations are solved numerically using the Newton-Krylov iterative method.

### 3.6.11 Values Between DAE and Models

ANDES adopts a decentralized architecture which provides each model a copy of variable values before equation evaluation. This architecture allows to parallelize the equation evaluation (in theory, or in practice if one works round the Python GIL). However, this architecture requires a coherent protocol for updating the DAE arrays and the *BaseVar* arrays. More specifically, how the variable and equations values from model *VarBase* should be summed up or forcefully set at the DAE arrays needs to be defined.

The protocol is relevant when a model defines subclasses of *BaseVar* that are supposed to be "public". Other models share this variable with *ExtAlgeb* or *ExtState*.

By default, all *v* and *e* at the same address are summed up. This is the most common case, such as a Bus connected by multiple devices: power injections from devices should be summed up.

In addition, *BaseVar* provides two flags, *v\_setter* and *e\_setter*, for cases when one *VarBase* needs to overwrite the variable or equation values.

### 3.6.12 Flags for Value Overwriting

*BaseVar* have special flags for handling value initialization and equation values. This is only relevant for public or external variables. The *v\_setter* is used to indicate whether a particular *BaseVar* instance sets the initial value. The *e\_setter* flag indicates whether the equation associated with a *BaseVar* sets the equation value.

The *v\_setter* flag is checked when collecting data from models to the numerical DAE array. If *v\_setter* is *False*, variable values of the same address will be added. If one of the variable or external variable has *v\_setter* is *True*, it will, at the end, set the values in the DAE array to its value. Only one *BaseVar* of the same address is allowed to have *v\_setter == True*.

### 3.6.13 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 be allowed to override the voltage initial value with the voltage set point.

In *Bus.\_\_init\_\_()*, one has

```
self.v = Algeb(v_str='1')
```

In *PV.\_\_init\_\_*, one can use

```
self.v0 = Param()
self.bus = IdxParam(model='Bus')

self.v = ExtAlgeb(src='v',
                  model='Bus',
                  indexer=self.bus,
                  v_str='v0',
                  v_setter=True)
```

where an *ExtAlgeb* is defined to access *Bus.v* using indexer *self.bus*. The *v\_str* line sets the initial value to *v0*. In the variable initialization phase for *PV*, *PV.v.v* is set to *v0*.

During the 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*.

## 3.7 Services

Services are helper variables outside the DAE variable list. Services are most often used for storing intermediate constants but can be used for special operations to work around restrictions in the symbolic framework. Services are value providers, meaning each service has an attribute *v* for storing service values. The base class of services is :py:mod` BaseService`, and the supported services are listed as follows.

---

*BaseService*([name, tex\_name, unit, info, Base class for Service.  
vtype])

*OperationService*([name, tex\_name, info]) Base class for a type of Service which performs specific operations.

---

### 3.7.1 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

---

### BaseService.assign\_memory

`BaseService.assign_memory(n)`

Assign memory for `self.v` and set the array to zero.

#### Parameters

##### **n**

[int] Number of elements of the value array. Provided by caller (Model.list2array).

**BaseService.get\_names****BaseService.get\_names()**Return *name* in a list**Returns****list**

A list only containing the name of the service variable

**Attributes**

<i>class_name</i>	Return the class name
<i>n</i>	Return the count of values in self.v.

**BaseService.class\_name****property BaseService.class\_name**

Return the class name

**BaseService.n****property BaseService.n**

Return the count of values in self.v.

Needs to be overloaded if v of subclasses is not a 1-dimensional array.

**Returns****int**

The count of elements in this variable

**3.7.2 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**

<code>assign_memory(n)</code>	Assign memory for <code>self.v</code> and set the array to zero.
<code>get_names()</code>	Return <i>name</i> in a list

**OperationService.assign\_memory**

`OperationService.assign_memory(n)`

Assign memory for `self.v` and set the array to zero.

**Parameters****n**

[int] Number of elements of the value array. Provided by caller (Model.list2array).

**OperationService.get\_names**

`OperationService.get_names()`

Return *name* in a list

**Returns****list**

A list only containing the name of the service variable

**Attributes**

<code>class_name</code>	Return the class name
<code>n</code>	Return the count of values in <code>self.v</code> .
<code>v</code>	Return values stored in <code>self._v</code> .

**OperationService.class\_name****property** OperationService.class\_name

Return the class name

**OperationService.n****property** OperationService.n

Return the count of values in self.v.

Needs to be overloaded if v of subclasses is not a 1-dimensional array.

**Returns****int**

The count of elements in this variable

**OperationService.v****property** OperationService.v

Return values stored in self.\_v. May be overloaded by subclasses.

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	The service type for reducing linear 2-D arrays into 1-D arrays
NumRepeat	The service type for repeating a 1-D array to linear 2-D arrays
IdxRepeat	The service type for repeating a 1-D list to linear 2-D list
EventFlag	Service type for flagging 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	Finds or creates devices linked to the given devices
BackRef	Collects idx-es for the backward references
RefFlatten	Converts BackRef list of lists into a 1-D list
InitChecker	Checks initial values against typical values
FlagValue	Flags values that equals the given value
Replace	Replace values that returns True for the given lambda func

### 3.7.3 Internal Constants

The most commonly used service is *ConstService*. It is used to store an array of constants, whose value is evaluated from a provided symbolic string. They are only evaluated once in the model initialization phase, ahead of variable initialization. *ConstService* comes handy when one wants to calculate intermediate constants from parameters.

For example, a turbine governor has a *NumParam R* for the droop. *ConstService* allows to calculate the inverse of the droop, the gain, and use it in equations. The snippet from a turbine governor's `__init__()` may look like

```
self.R = NumParam()
self.G = ConstService(v_str='u/R')
```

where *u* is the online status parameter. The model can thus 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

```
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 not solved with other algebraic equations, meaning that there is one step "delay" between the algebraic variables and `VarService`. Use an algebraic variable whenever possible.

### Examples

In ESST3A model, the voltage and current sensors ( $v_d + jv_q$ ), ( $I_d + jI_q$ ) 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))', )
```

```
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*.

### 3.7.4 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. *ExtService* is used for this purpose. In the `__init__()` of a synchronous generator model, one can define the following to retrieve *StaticGen.p* as *p0*:

```
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 GENCLSMModel(Model):
    def __init__(...):
        ...
        self.p0 = ExtService(src='p',
                             model='StaticGen',
                             indexer=self.gen,
                             tex_name='P_0')
```

## 3.7.5 Shape Manipulators

This section is for advanced model developer.

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 will use such indices to retrieve variable addresses. The retrieved addresses usually has a different length of the referencing model and cannot be used directly for calculation. Shape manipulator services can be used in such case.

*NumReduce* is a helper Service type which reduces a linearly stored 2-D ExtParam into 1-D Service. *NumRepeat* is a helper Service type which 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.ACTopology = BackRef()* to collect devices in the *ACTopology* 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  $V_n$  in one Area. In the `Area` class, one defines

```
class AreaModel(...):
    def __init__(...):
        ...
        # backward reference from `Bus`
        self.Bus = BackRef()

        # collect the  $V_n$  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	$V_n$
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,
                  )
```

## 3.7.6 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.

## 3.7.7 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 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.

```
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.ACTopology = BackRef()* to collect devices in the *ACTopology* 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.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,
                  )
```

## 3.7.8 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.

### 3.7.9 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 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  $\geq$  the given value element-wise.

Parameters that satisfy the comparison ( $u >$  or  $\geq$  value) will be 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  $\leq$  the given value element-wise.

Parameters that satisfy the comparison ( $u <$  or  $\leq$  value) will be 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).

## 3.7.10 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)
```

### 3.7.11 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.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.

```
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.



```
class andes.core.service.RandomService (func=<built-in method rand of  
numpy.random.mtrand.RandomState object>,  
**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.

## 3.8 Discrete

### 3.8.1 Background

The discrete component library contains a special type of block for modeling the discontinuity in power system devices. Such continuities can be device-level physical constraints or algorithmic limits imposed on controllers.

The base class for discrete components is *andes.core.discrete.Discrete*.

<i>Discrete</i> ([ <i>name</i> , <i>tex_name</i> , <i>info</i> , <i>no_warn</i> , ...])	Base discrete class.
---	----------------------

## 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

<code>check_eq(**kwargs)</code>	This function is called in <code>l_check_eq</code> after updating equations.
<code>check_iter_err([niter, err])</code>	Check if the minimum iteration or maximum error is reached so that this discrete block should be enabled.
<code>check_var(*args, **kwargs)</code>	This function is called in <code>l_update_var</code> before evaluating equations.
<code>get_names()</code>	Available symbols from this class
<code>get_tex_names()</code>	Return <code>tex_names</code> of exported flags.
<code>get_values()</code>	
<code>list2array(n)</code>	Allocate memory for the discrete flags specified in <code>self.export_flags</code> .
<code>warn_init_limit()</code>	Warn if associated variables are initialized at limits.

#### Discrete.check\_eq

```
Discrete.check_eq(**kwargs)
```

This function is called in `l_check_eq` after updating equations.

It updates internal flags, set differential equations, and record pegged variables.

#### Discrete.check\_iter\_err

```
Discrete.check_iter_err(niter=None, err=None)
```

Check if the minimum iteration or maximum error is reached so that this discrete block should be enabled.

Only when both `niter` and `err` are given, (`niter < min_iter`) , and (`err > err_tol`) it will return False.

This logic will start checking the discrete states if called from an external solver that does not feed `niter` or `err` at each step.

**Returns****bool**

True if it should be enabled, False otherwise

**Discrete.check\_var**

```
Discrete.check_var(*args, **kwargs)
```

This function is called in `l_update_var` before evaluating equations.

It should update internal flags only.

**Parameters****adjust\_upper**

[bool] True to adjust the upper limit to the value of the variable. Supported by limiters.

**adjust\_lower**

[bool] True to adjust the lower limit to the value of the variable. Supported by limiters.

**Discrete.get\_names**

```
Discrete.get_names()
```

Available symbols from this class

**Returns****Discrete.get\_tex\_names**

```
Discrete.get_tex_names()
```

Return tex\_names of exported flags.

TODO: Fix the bug described in the warning below.

**Returns****list**

A list of tex\_names for all exported flags.

**Warning:** If underscore `_` appears in both flag `tex_name` and `self.tex_name` (for example, when this discrete is within a block), the exported `tex_name` will become invalid for SymPy. Variable name substitution will fail.

## Discrete.get\_values

```
Discrete.get_values()
```

## Discrete.list2array

```
Discrete.list2array(n)
```

Allocate memory for the discrete flags specified in *self.export\_flags*.

### Parameters

**n**

[int] Number of elements in the array. Provided by the calling function.

## Discrete.warn\_init\_limit

```
Discrete.warn_init_limit()
```

Warn if associated variables are initialized at limits.

## Attributes

---

```
class_name
```

---

## Discrete.class\_name

**property** Discrete.class\_name

The uniqueness of discrete components is the way it works. Discrete components take inputs, criteria, and exports a set of flags with the 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, two v-providers as the upper and the lower bound. It exports three flags: *zi* (within bound), *zl* (below lower bound), and *zu* (above upper bound). See the code example in *models/pv.py* for an example voltage-based PQ-to-Z conversion.

It is important to note when the flags are updated. Discrete subclasses can use three methods to check and update the value and equations. Among these methods, *check\_var* is called *before* equation evaluation, but *check\_eq* and *set\_eq* are called *after* equation update. 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.

ANDES includes the following types of discrete components.

### 3.8.2 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 ( $\geq$  or  $\leq$ ).

##### **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

## Notes

If not enabled, the default flags are `zu = zl = 0, zi = 1`.

### 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

```
class andes.core.discrete.SortedLimiter(u, lower, upper, n_select: int = 5, name=None,
                                         tex_name=None, enable=True,
                                         absViolation=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  $\text{abs}(\text{violation}/\text{limit})$  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

## 3.8.3 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 piecewise 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 IEEEEST 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, 0, 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.

## 3.8.4 Deadband

```
class andes.core.discrete.DeadBand(u, center, lower, upper, enable=True, equal=False,
                                zu=0.0, zl=0.0, zi=0.0, 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=selfdbc,
                    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'
```

## 3.8.5 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

## 3.9 Blocks

### 3.9.1 Background

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. The base class for blocks is `andes.core.block.Block`.

The supported blocks include `Lag`, `LeadLag`, `Washout`, `LeadLagLimit`, `PIController`. In addition, the base class for piece-wise nonlinear functions, `Piecewise` is provided. `Piecewise` is used for implementing the quadratic saturation function `MagneticQuadSat` and exponential saturation function `MagneticExpSat`.

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 placed in a dictionary, `self.vars` at the end of the block constructor.

Blocks can be nested as advanced usage. See the following API documentation 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'
```

### 3.9.2 Transfer Functions

The following transfer function blocks have been implemented. They can be imported to build new models.

#### Linear

```
class andes.core.block.Gain(u, K, name=None, tex_name=None, info=None)
```

Gain block.



Exports an algebraic output `y`.

```
define()
```

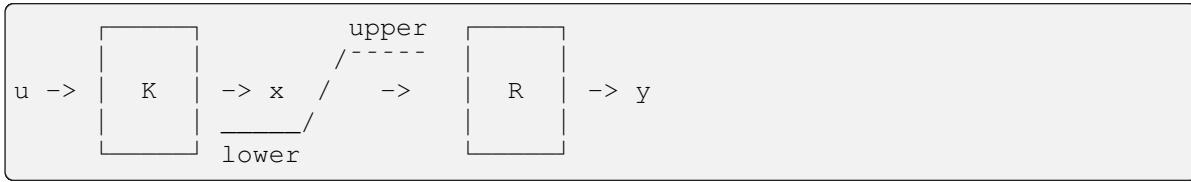
Implemented equation and the initial condition are

$$\begin{aligned}y &= Ku \\y^{(0)} &= Ku^{(0)}\end{aligned}$$

```
class andes.core.block.GainLimiter(u, K, R, lower, upper, no_lower=False,
                                    no_upper=False, sign_lower=1, sign_upper=1,
                                    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`.



## 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,  $[x_0, x_1, \dots, x_{n-1}]$  to define  $N+1$  ranges, namely  $(-\infty, x_0)$ ,  $(x_0, x_1)$ , ...,  $(x_{n-1}, +\infty)$ . and a list of  $N+1$  function strings  $[fun_0, \dots, fun_n]$ .

Inputs that fall within each range applies the corresponding function. The first range  $(-\infty, x_0)$  applies  $fun_0$ , and the last range  $(x_{n-1}, +\infty)$  applies the last function  $fun_n$ .

The function returns zero if no condition is met.

## 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.

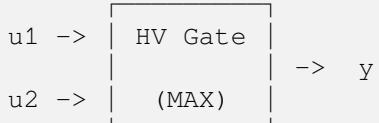
### **define()**

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.



```
class andes.core.block.LVGate(u1, u2, name=None, tex_name=None, info=None)
```

**Low Value Gate.** Outputs the minimum of the two inputs.



```
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).

## 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



## First Order

Integrator block.



Exports a differential variable  $y$ .

The initial output needs to be specified through  $y0$ .

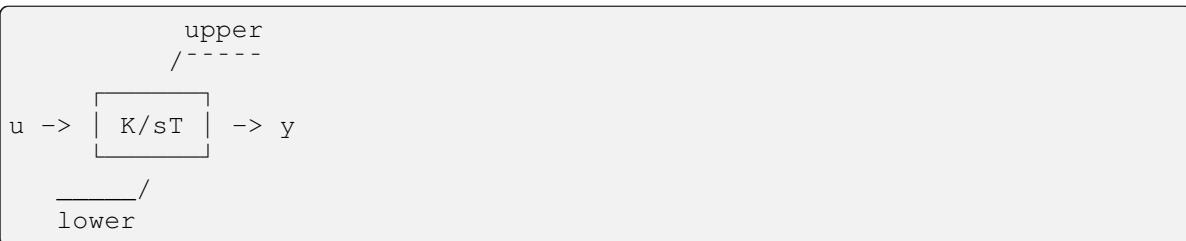
**define()**

Implemented equation and the initial condition are

$$\begin{aligned}\dot{y} &= Ku \\ y^{(0)} &= 0\end{aligned}$$

```
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.



Exports a differential variable  $y$  and an AntiWindup *lim*. The initial output must be specified through  $y0$ .

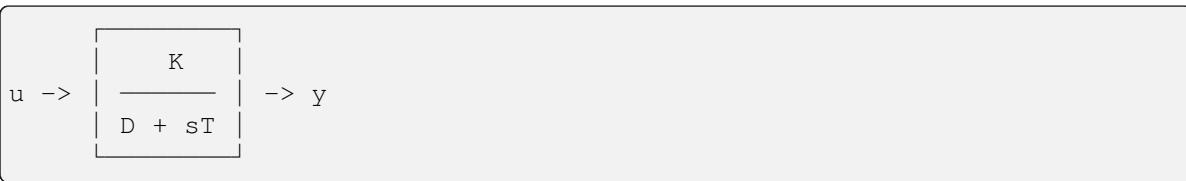
**define()**

Implemented equation and the initial condition are

$$\begin{aligned}\dot{y} &= Ku \\ y^{(0)} &= 0\end{aligned}$$

```
class andes.core.block.Lag(u, T, K, D=1, name=None, tex_name=None, info=None)
```

Lag (low pass filter) transfer function.



Exports one state variable  $y$  as the output.

### Parameters

**K**

Gain

**T**

Time constant

**D**

Constant

**u**  
Input variable

**define()**

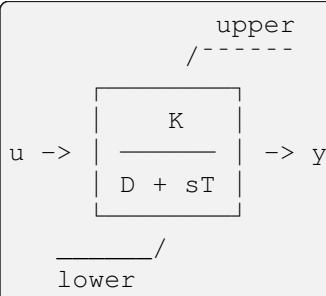
## Notes

Equations and initial values are

$$\begin{aligned} Ty &= (Ku - Dy) \\ y^{(0)} &= Ku/D \end{aligned}$$

```
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.



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

$$\begin{aligned} T\dot{y} &= (Ku - Dy) \\ y^{(0)} &= Ku/D \end{aligned}$$

```
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

$$\begin{aligned} T\dot{y} &= (1 - \text{freeze}) * (Ku - y) \\ y^{(0)} &= Ku \end{aligned}$$

```
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

$$\begin{aligned} T\dot{y} &= (1 - \text{freeze})(Ku - y) \\ y^{(0)} &= Ku \end{aligned}$$

*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.



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

$$Ty = (Ku - y)$$

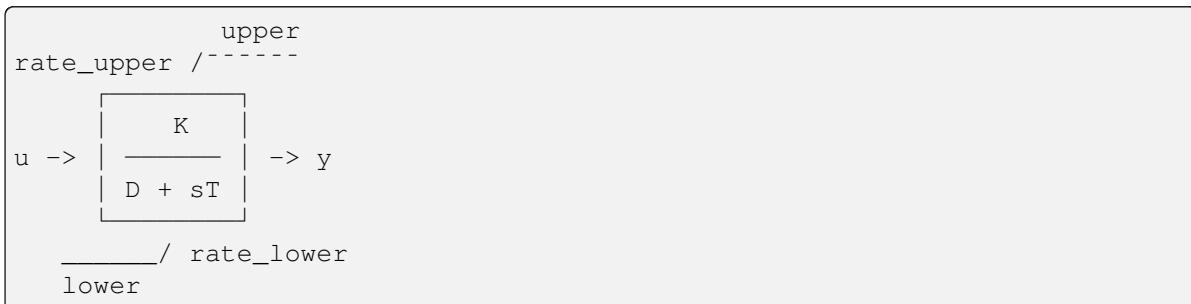
$$y^{(0)} = Ku$$

```

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.



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 ()**

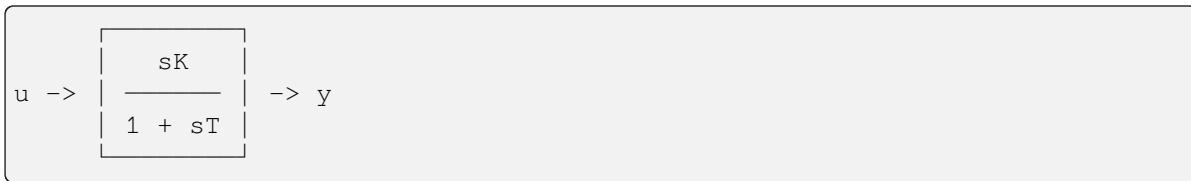
### Notes

Equations and initial values are

$$\begin{aligned} T\dot{y} &= (Ku - Dy) \\ y^{(0)} &= Ku/D \end{aligned}$$

**class** andes.core.block.**Washout** ( $u, T, K, name=None, tex\_name=None, info=None$ )

Washout filter (high pass) block.



Exports state  $x$  (symbol  $x'$ ) and output algebraic variable  $y$ .

**define ()**

### Notes

Equations and initial values:

$$\begin{aligned} \dot{x'} &= (u - x') \\ Ty &= K(u - x') \\ x'^{(0)} &= u \\ y^{(0)} &= 0 \end{aligned}$$

---

```
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

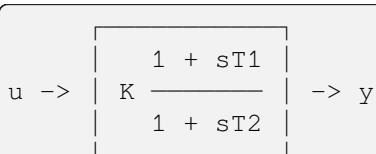
Equations and initial values:

$$\begin{aligned} \dot{Tx'} &= (u - x') \\ Ty &= z_0 K(u - x') + z_1 Tx \\ x'^{(0)} &= u \\ y^{(0)} &= 0 \end{aligned}$$

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.



Exports two variables: internal state  $x$  and output algebraic variable  $y$ .

### Parameters

#### **T1**

[BaseParam] Time constant 1

#### **T2**

[BaseParam] Time constant 2

#### **zero\_out**

[bool] True to allow zeroing out lead-lag as a pass through (when  $T1=T2=0$ )

## Notes

To allow zeroing out lead-lag as a pure gain, set `zero_out` to `True`.

```
define()
```

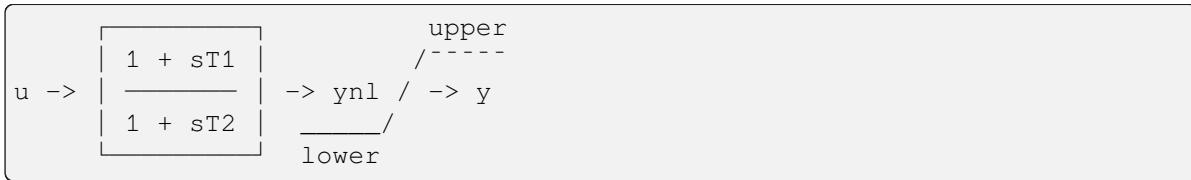
## Notes

Implemented equations and initial values

$$\begin{aligned} T_2 \dot{x}' &= (u - x') \\ T_2 y &= K T_1 (u - x') + K T_2 x' + E_2, \text{ where} \\ E_2 &= \begin{cases} (y - Kx') & \text{if } T_1 = T_2 = 0 \& zero\_out = True \\ 0 & \text{otherwise} \end{cases} \\ x'^{(0)} &= u \\ y^{(0)} &= Ku \end{aligned}$$

```
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).



Exports four variables: state  $x$ , output before hard limiter  $ynl$ , output  $y$ , and AntiWindup  $lim$ .

```
define()
```

## Notes

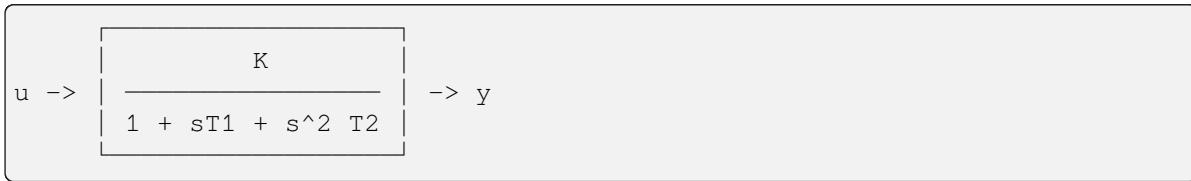
Implemented control block equations (without limiter) and initial values

$$\begin{aligned} T_2 \dot{x}' &= (u - x') \\ T_2 y &= T_1 (u - x') + T_2 x' \\ x'^{(0)} &= y^{(0)} = u \end{aligned}$$

## Second Order

**class** andes.core.block.**Lag2ndOrd**(*u, K, T1, T2, name=None, tex\_name=None, info=None*)

Second order lag transfer function (low-pass filter).



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()**

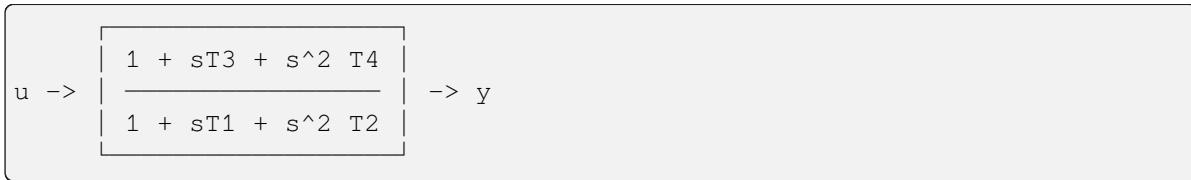
### Notes

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.



Exports two internal states (*x1* and *x2*) and output algebraic variable *y*.

The current implementation allows any or all parameters to be zero. Four LessThan blocks are used to check if the parameter values are all zero. If yes, *y* = *u* will be imposed in the algebraic equation.

---

```
define()
```

## Notes

Implemented equations and initial values are

$$\begin{aligned} T_2 \dot{x}_1 &= u - x_2 - T_1 x_1 \\ \dot{x}_2 &= x_1 \\ T_2 y &= T_2 x_2 + T_2 T_3 x_1 + T_4(u - x_2 - T_1 x_1) + E_2, \text{ where} \\ E_2 &= \begin{cases} (y - x_2) & \text{if } T_1 = T_2 = T_3 = T_4 = 0 \& zero\_out = True \\ 0 & \text{otherwise} \end{cases} \\ x_1^{(0)} &= 0 \\ x_2^{(0)} &= y^{(0)} = u \end{aligned}$$

## 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

$$\begin{aligned} \dot{x}_i &= k_i * (u - ref) \\ y &= x_i + k_p * (u - ref) \end{aligned}$$

```
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  $x_i$  and one algebraic variable  $y$  are added.

Equations implemented are

$$\begin{aligned}\dot{x}_i &= k_i * (u - ref) \\ y &= x_i + k_p * (u - ref)\end{aligned}$$

```
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  $x_i$  and one algebraic variable  $y$  are added.

Equations implemented are

$$\begin{aligned}\dot{x}_i &= k_i * (u - ref) \\ y &= (1 - \text{freeze}) * (x_i + k_p * (u - ref)) + \text{freeze} * y\end{aligned}$$

```
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.

$$u \rightarrow \left[ \frac{ki}{kp + \frac{s}{s + Td}} + \frac{skd}{1 + sTd} \right] \rightarrow y$$

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 suggested 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

**define()**

Define equations for the PID Controller.

## Notes

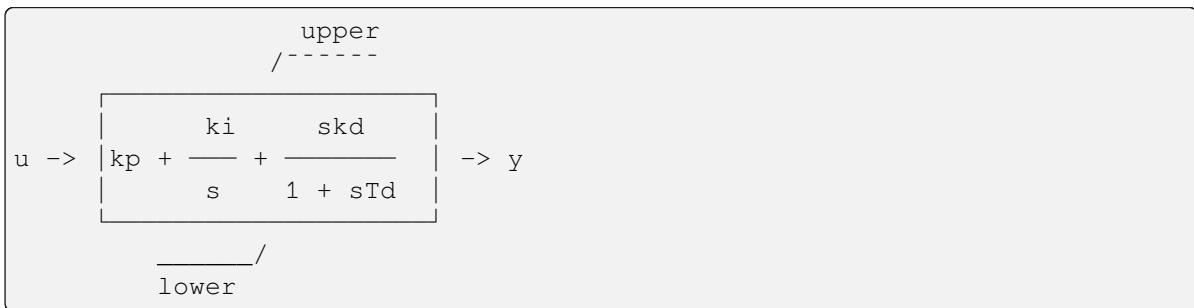
One PIController PIC, one Washout xd, and one algebraic variable y are added.

Equations implemented are

$$\begin{aligned}\dot{x}_i &= k_i * (u - ref) \\ xd &= Washout(u - ref)y \\ &= x_i + k_p * (u - ref) + xd\end{aligned}$$

```
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.



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 suggested 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  $x_i$  and one algebraic variable  $y$  are added.

Equations implemented are

$$\begin{aligned} \dot{x}_i &= k_i * (u - ref) \\ y &= x_i + k_p * (u - ref) \end{aligned}$$

```
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'
```

### 3.9.3 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

$$E_1 S_{E1} = A e^{E_1 \times B} \quad E_2 S_{E2} = A e^{E_2 \times B}$$

The solutions are given by

$$E_1 S_{E1} e^{\frac{E_1 \log\left(\frac{E_2 S_{E2}}{E_1 S_{E1}}\right)}{E_1 - E_2}} - \frac{\log\left(\frac{E_2 S_{E2}}{E_1 S_{E1}}\right)}{E_1 - E_2}$$

### 3.9.4 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 `W01` and `W02`. 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.

Naming conventions are not strictly enforced. Expressiveness and concision are encouraged.

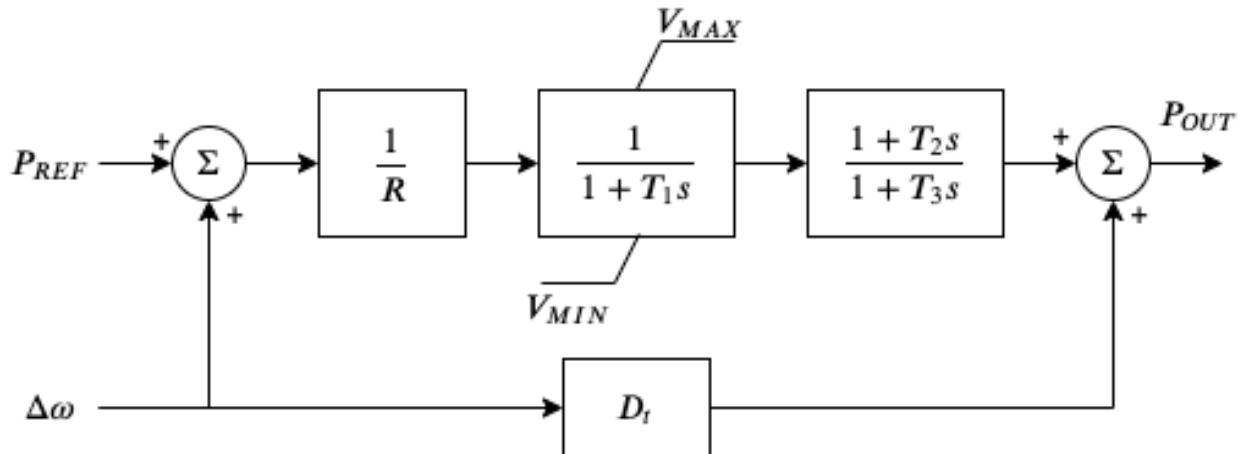
## 3.10 Examples

We show two examples to demonstrate modeling from equations and modeling from control block diagrams.

- The `TGOV1` example shows code snippet for equation-based modeling and, as well as code for block-based modeling.
- The `IEEEEST` example walks through the source code and explains the complete setup, including optional parameters, input selection, and manual per-unit conversion.

### 3.10.1 TGOV1

The `TGOV1` turbine governor model is shown as a practical example using the library.



This model is composed of a lead-lag transfer function and a first-order lag transfer function with an anti-windup limiter, which are sufficiently complex for demonstration. The corresponding differential equations

and algebraic equations are given below.

$$\begin{bmatrix} \dot{x}_{LG} \\ \dot{x}_{LL} \end{bmatrix} = \begin{bmatrix} z_{i,lim}^{LG} (P_d - x_{LG}) / T_1 \\ (x_{LG} - x_{LL}) / T_3 \end{bmatrix}$$

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} (1 - \omega) - \omega_d \\ R \times \tau_{m0} - P_{ref} \\ (P_{ref} + \omega_d) / R - P_d \\ D_t \omega_d + y_{LL} - P_{OUT} \\ \frac{T_2}{T_3} (x_{LG} - x_{LL}) + x_{LL} - y_{LL} \\ u (P_{OUT} - \tau_{m0}) \end{bmatrix}$$

where *LG* and *LL* denote the lag block and the lead-lag block,  $\dot{x}_{LG}$  and  $\dot{x}_{LL}$  are the internal states,  $y_{LL}$  is the lead-lag output,  $\omega$  the generator speed,  $\omega_d$  the generator under-speed,  $P_d$  the droop output,  $\tau_{m0}$  the steady-state torque input, and  $P_{OUT}$  the turbine output that will be summed at the generator.

The code to describe the above model using equations is given below. The complete code can be found in class `TGOV1ModelAlt` in `andes/models/governor.py`.

```
def __init__(self, system, config):
    # 1. Declare parameters from case file inputs.
    self.R = NumParam(info='Turbine governor droop',
                      non_zero=True, ipower=True)
    # Other parameters are omitted.

    # 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',
                       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',
```

(continues on next page)

(continued from previous page)

```

        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')

```

Another implementation of *TGOVI* makes extensive use of the modeling blocks. The resulting code is more readable as follows.

```

def __init__(self, system, config):
    TGBase.__init__(self, system, config)

    self.gain = ConstService(v_str='u/R')

    self.pref = Algeb(info='Reference power input',
                      tex_name='P_{ref}',
                      v_str='tm0 * R',
                      e_str='tm0 * R - pref',
                      )

    self.wd = Algeb(info='Generator under speed',
                    unit='p.u.',
                    tex_name=r'\omega_{dev}',
                    v_str='0',
                    e_str='(wref - omega) - wd',
                    )

    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')

    self.LAG = LagAntiWindup(u=self.pd,
                            K=1,
                            T=self.T1,
                            lower=self.VMIN,
                            upper=self.VMAX,
                            )

    self.LL = LeadLag(u=self.LAG_y,
                      T1=self.T2,
                      T2=self.T3,
                      )

```

(continues on next page)

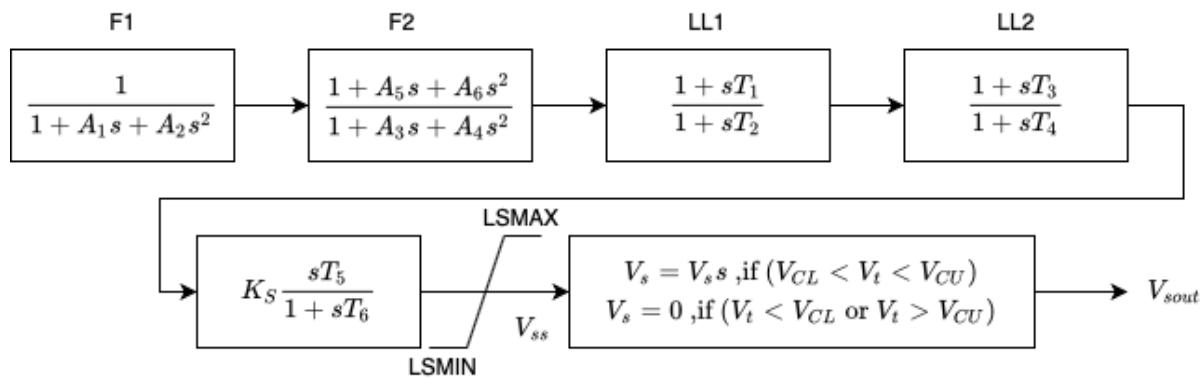
(continued from previous page)

```
self.pout.e_str = '(LL_y + Dt * wd) - pout'
```

The complete code can be found in class `TGOV1Model` in `andes/models/governor.py`.

### 3.10.2 IEEEEST

In this example, we will explain step-by-step how `IEEEEST` is programmed. The block diagram of IEEEEST is given as follows. We recommend you to open up the source code in `andes/models/pss.py` and then continue reading.



First of all, modeling components are imported at the beginning.

Next, `PSSBaseData` is defined to hold parameters shared by all PSSs. `PSSBaseData` inherits from `ModelData` and calls the base constructor. There is only one field `avr` defined for the linked exciter idx.

Then, `IEEEESTData` defines the input parameters for IEEEEST. Use `IdxParam` for fields that store idx-es of devices that IEEEEST devices link to. Use `NumParam` for numerical parameters.

### PSSBase

`PSSBase` is defined for the common (external) parameters, services and variables shared by all PSSs. The class and constructor signatures are

```
class PSSBase(Model):
    def __init__(self, system, config):
        super().__init__(system, config)
```

`PSSBase` inherits from `Model` and calls the base constructor. Note that the call to `Model`'s constructor takes two positional arguments, `system` and `config` of types `System` and `ModelConfig`. Next, the group is specified, and the model flags are set.

**Note:** It is important to set the `TDS` flag to register the model. If not set, data for the model will be successfully loaded, but the variables in the model will not receive any address, and the model equations will be skipped.

There is a similar flag `self.flags.pflow` for models to participate in power flow calculations. Most dynamic models, however, are initialized after power flow.

```
self.group = 'PSS'
self.flags.tds = True
```

Next, `Replace` is used to replace input parameters that satisfy a lambda function with new values.

```
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)
```

The value replacement happens when `VCUr` and `VCLR` is first accessed. `Replace` is executed in the model initialization phase (at the end of services update).

Next, the indices of connected generators, buses, and bus frequency measurements are retrieved. Synchronous generator `idx` is retrieved with

```
self.syn = ExtParam(model='Exciter', src='syn', indexer=self.avr,
                     ↪export=False,
                     info='Retrieved generator idx', vtype=str)
```

Using the retrieved `self.syn`, it retrieves the buses to which the generators are connected.

```
self.bus = ExtParam(model='SynGen', src='bus', indexer=self.syn, export=False,
                     info='Retrieved bus idx', vtype=str, default=None,
                     )
```

PSS models support an optional remote bus specified through parameter `busr`. When `busr` is `None`, the generator-connected bus should be used. The following code uses `DataSelect` to select `busr` if available but falls back to `bus` otherwise.

```
self.buss = DataSelect(self.busr, self.bus, info='selected bus (bus or busr)')
```

Each PSS links to a bus frequency measurement device. If the input data does not specify one or the specified one does not exist, `DeviceFinder` can find the correct measurement device for the bus where frequency measurements should be taken.

```
self.busfreq = DeviceFinder(self.busf, link=self.buss, idx_name='bus')
```

where `busf` is the optional frequency measurement device `idx`, `buss` is the bus `idx` for which measurement device needs to be found or created.

Next, external parameters, variables and services are retrieved. Note that the PSS output `vsout` is pre-allocated but the equation string is left to specific models.

## IEEEESTModel

IEEEESTModel inherits from PSSBase and adds specific model components. After calling PSSBase's constructor, IEEEESTModel adds config entries to allow specifying the model for frequency measurement, because there may be multiple frequency measurement models in the future.

```
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',)})
```

We set the chosen measurement model to busf so that DeviceFinder knows which model to use if it needs to create new devices.

```
self.busf.model = self.config.freq_model
```

Next, because bus voltage is an algebraic variable, we use Derivative to calculate the finite difference to approximate its derivative.

```
self.dv = Derivative(self.v, tex_name='dV/dt', info='Finite difference of bus_
→voltage')
```

Then, we retrieve the coefficient to convert power from machine base to system base using ConstService, given by Sb / Sn. This is needed for input mode 3, electric power in machine base.

```
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 that the ExtService access the pu\_coeff field of the M variables of synchronous generators. Since M is a machine-base power quantity, M.pu\_coeff stores the multiplication coefficient to convert each of them from machine bases to the system base, which is Sb / Sn.

The input mode is parsed into boolean flags using Switcher:

```
self.SW = Switcher(u=self.MODE,
                    options=[0, 1, 2, 3, 4, 5, 6],
                    )
```

where the input u is the MODE parameter, and options is a list of accepted values. Switcher boolean arrays s0, s1, ..., sN, where N = len(options) - 1. We added 0 to options for padding so that SW\_s1 corresponds to MODE 1. It improves the readability of the code as we will see next.

The input signal sig is an algebraic variable given by

```
self.sig = Algeb(tex_name='S_{sig}', info='Input signal',
                  )
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'
```

(continues on next page)

(continued from previous page)

```
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'
```

The `v_str` and `e_str` are separated from the constructor to improve readability. They construct piece-wise functions to select the correct initial values and equations based on mode. For any variables in `v_str`, they must be defined before `sig` so that they will be initialized ahead of `sig`. Clearly, `omega`, `tm`, and `v` are defined in `PSSBase` and thus come before `sig`.

The following comes the most effective part: modeling using transfer function blocks. We utilized several blocks to describe the model from the diagram. Note that the output of a block is always the block name followed by `_y`. For example, the input of `F2` is the output of `F1`, given by `F1_y`.

```
self.F1 = Lag2ndOrd(u=self.sig, K=1, T1=self.A1, T2=self.A2)

self.F2 = LeadLag2ndOrd(u=self.F1_y, T1=self.A3, T2=self.A4,
                        T3=self.A5, T4=self.A6, zero_out=True)

self.LL1 = LeadLag(u=self.F2_y, T1=self.T1, T2=self.T2, zero_out=True)

self.LL2 = LeadLag(u=self.LL1_y, T1=self.T3, T2=self.T4, zero_out=True)

self.Vks = Gain(u=self.LL2_y, K=self.KS)

self.WO = WashoutOrLag(u=self.Vks_y, T=self.T6, K=self.T5, name='WO',
                      zero_out=True) # WO_y == Vss

self.VLIM = Limiter(u=self.WO_y, lower=self.LSMIN, upper=self.LSMAX,
                     info='Vss 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 * LMIN -_
                Vss')

self.OLIM = Limiter(u=self.v, lower=self.VCLr, upper=self.VCUR,
                     info='output limiter')

self.vsout.e_str = 'OLIM_zi * Vss - vsout'
```

In the end, the output equation is assigned to `vsout.e_str`. It completes the equations of the IEEEEST model.

## Finalize

Assemble IEEEESTData and IEEEESTModel into IEEEEST:

```
class IEEEEST(IEEEESTData, IEEEESTModel):
    def __init__(self, system, config):
        IEEEESTData.__init__(self)
        IEEEESTModel.__init__(self, system, config)
```

Locate and edit the file andes/models/\_\_init\_\_.py. The variable file\_classes is a list of tuples that stores the source file and class names in strings. In each tuple, the first element is the .py file name in the models folder, and the second element is a list of class names to be imported from that file.

Find the line with pss, then add IEEEEST to the corresponding list of model names. The line will look like

```
file_classes = list([
    ...
    ('pss', ['IEEEEST', 'ST2CUT']),
    ...
])
```

Note in the above that the string 'IEEEEST' is used. The line above is valid as long as from andes.models.pss import IEEEEST is valid. If the source file name does not exist in any line of file\_classes, one may add it after all prerequisite models. For example, the pss line should be added after exciters (and generators, of course).

Finally, locate andes/models/group.py, check if the class PSS exist. It needs to match the group name of IEEEEST. If not, create one by inheriting from GroupBase:

```
class PSS(GroupBase):
    """Power system stabilizer group."""

    def __init__(self):
        super().__init__()
        self.common_vars.extend(['vsout',])
```

where we added vsout to the common\_vars list. All models in the PSS group must have a variable named vsout, which is defined in PSSBase.

This completes the IEEEEST model. When developing new models, use andes prepare to generate numerical code and start debugging.

---

**CHAPTER  
FOUR**

---

**RELEASE NOTES**

The APIs before v3.0.0 are in beta and may change without prior notice.

## 4.1 v1.9 Notes

### 4.1.1 v1.9.0 (2023-12-07)

- 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.

## 4.2 v1.8 Notes

### 4.2.1 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.

#### **4.2.2 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.

#### **4.2.3 v1.8.8 (2023-04-24)**

- Repository transferred to [github.com/curent/andes](https://github.com/curent/andes).
- Fix for frequency-dependent load model `FLoad` by not exporting the external `bus` parameter.

#### **4.2.4 v1.8.7 (2023-03-10)**

- Added IEEE 39-bus test case with dynamics in the ANDES xlsx format. Contributed by @jiningwang.
- In `interop.pandapower`, the `_verify_pf` function will not be called if TDS has been initialized. This prevents the breaking of variable addresses.

#### **4.2.5 v1.8.6 (2023-02-13)**

- Minor fix in `timeseries.py` for backward compatibility with Python 3.6.

#### **4.2.6 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.

#### **4.2.7 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.

## 4.2.8 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.

## 4.2.9 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.

## 4.2.10 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.

## 4.2.11 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.

## 4.3 v1.7 Notes

### 4.3.1 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.

### **4.3.2 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

### **4.3.3 v1.7.6 (2022-07-11)**

- In eigenvalue analysis, added parameter sweeping and scatter plot for root loci. See revised Example 4.
- Documentation improvements.

### **4.3.4 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.

### **4.3.5 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.

### 4.3.6 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.

### 4.3.7 v1.7.2 (2022-06-07)

- Improved documentation and examples.

### 4.3.8 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`.

### 4.3.9 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.

## 4.4 v1.6 Notes

### 4.4.1 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.

### 4.4.2 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()`.

#### 4.4.3 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 `REGCPI` model for generic converters with PLL support.
- Fixed PSS/E parser for `HYGOV`.

#### 4.4.4 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.

#### **4.4.5 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`. See [\*Extra support package\*](#) for more information.
- 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`.

#### **4.4.6 v1.6.1 (2022-03-13)**

- Revamped documentation with a much improved "Getting started" section.

#### **4.4.7 v1.6.0 (2022-03-11)**

- Migrated documentation to the pydata template.
- Added compatibility with SymPy 1.9 and 1.10.

### **4.5 v1.5 Notes**

#### **4.5.1 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.

## 4.5.2 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.

## 4.5.3 v1.5.10 (2022-02-01)

- Fixed one equation in *REGC\_A*.

## 4.5.4 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.

## 4.5.5 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 excitors to zeros. All `vout` equations need to be multiplied by `ue`.

## 4.5.6 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_upper`. 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.

#### **4.5.7 v1.5.6 (2021-11-25)**

- Allow specifying config options through command-line arguments `--config-option`.
- Added a voltage and frequency playback model PLBFU1.
- Bug fixes to an SEXS equation.

#### **4.5.8 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)

#### **4.5.9 v1.5.4 (2021-11-02)**

- Fixed a bug in generated `select` functions that omitted the coefficients of `__ones`.

#### 4.5.10 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 separated time-domain integration method into `daeint.py`.

#### 4.5.11 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.

#### 4.5.12 v1.5.1 (2021-10-23)

- Restored compatibility with SymPy 1.6.
- Added a group for voltage compensators.
- New models: `IEEEVC` and `GAST`.

#### 4.5.13 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.

## 4.6 v1.4 Notes

### 4.6.1 v1.4.4 (2021-10-05)

- Bug fixes for refreshing generated code.

### 4.6.2 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.

### 4.6.3 v1.4.2 (2021-09-12)

- Bug fixes
- Dropped support for `cvxoptklu`.

### 4.6.4 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.

## 4.6.5 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.

## 4.7 v1.3 Notes

### 4.7.1 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.

### 4.7.2 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.

### 4.7.3 v1.3.10 (2021-06-08)

- Bug fixes for controllers when generators are off.

#### **4.7.4 v1.3.9 (2021-06-02)**

- Bug fixes in excitors when generators are offline.
- Added `safe_div` function for initialization equations.

#### **4.7.5 v1.3.8 (2021-06-02)**

- Added REGCVSG model for voltage-source controlled renewables.
- Turbine governors are now aware of the generator connection status.

#### **4.7.6 v1.3.7 (2021-05-03)**

- Allow manually specifying variables needing initialization preceding a variable. Specify a list of variable names through `BaseVar.deps`.

#### **4.7.7 v1.3.6 (2021-04-23)**

- Patched ESD1 model. Converted `distributed.py` into a package.
- Bug fixes.

#### **4.7.8 v1.3.5 (2021-03-20)**

- Fixed a bug in connectivity check when bus 0 is islanded.
- Updated notebook examples.
- Updated tutorials.

#### **4.7.9 v1.3.4 (2021-03-13)**

- Fixed a bug for the generated renewable energy code.

#### **4.7.10 v1.3.2 (2021-03-08)**

- Relaxed the version requirements for NumPy and SymPy.

#### 4.7.11 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.

#### 4.7.12 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 `IEEEET1`, `EXAC4`.
- Refactored excitors, turbine governors, and renewable models into modules.

### 4.8 v1.2 Notes

#### 4.8.1 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.

#### 4.8.2 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 used as a factor for scaling algebraic equations for better convergence. Setting it to 1.0 functions the same as before.

### **4.8.3 v1.2.6 (2020-12-01)**

- Added *TGOVIN* 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).

### **4.8.4 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.

### **4.8.5 v1.2.4 (2020-11-13)**

- Added switched shunt class *ShuntSw*.
- BaseParam takes *inconvert* and *oconvert* for converting parameter elements from and to files.

### **4.8.6 v1.2.3 (2020-11-02)**

- Support variable *sys\_mva* (system base mva) in equation strings.
- Default support for KVXOPT through pip installation.

### **4.8.7 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.

#### 4.8.8 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.

#### 4.8.9 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`.

### 4.9 v1.1 Notes

#### 4.9.1 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.

## **4.9.2 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.

## **4.9.3 v1.1.3 (2020-09-05)**

- Improved documentation.
- Minor bug fixes.

## **4.9.4 v1.1.2 (2020-09-03)**

- Patched time-domain for continuing simulation.

## **4.9.5 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.

## **4.9.6 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*.

## 4.10 v1.0 Notes

### 4.10.1 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*.

### 4.10.2 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 *FlagNone* to *FlagValue* with an option to flip the flags.
- Other tweaks.

#### **4.10.3 v1.0.6 (2020-07-08)**

- Patched step size adjustment algorithm.
- Added Area Control Error (ACE) model.

#### **4.10.4 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.

#### **4.10.5 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.

#### **4.10.6 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.

#### **4.10.7 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.

#### **4.10.8 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.

## 4.10.9 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.

## 4.11 Pre-v1.0.0

### 4.11.1 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.

### 4.11.2 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.

#### **4.11.3 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_ieest.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`.

#### **4.11.4 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_ieest.xlsx` case, simulation time was reduced from 3.5s to 2.7s.

#### 4.11.5 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 IEEEEST stabilizer.
- Implemented COI for generator speed and angle measurement.

#### 4.11.6 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 IEEEEST stabilizer with some fixes needed.

#### 4.11.7 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.

#### **4.11.8 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;

#### **4.11.9 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.

#### **4.11.10 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.

#### **4.11.11 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  
FIVE**

---

## **MODEL REFERENCE**

Use the left navigation pane to locate the group and model and view details.

### Supported Groups and Models

Group	Models
<i>ACLine</i>	<i>Line</i>
<i>ACShort</i>	<i>Jumper</i>
<i>ACTopology</i>	<i>Bus</i>
<i>Calculation</i>	<i>ACE, ACEc, COI</i>
<i>Collection</i>	<i>Area</i>
<i>DCLink</i>	<i>Ground, R, L, C, RCp, RCs, RLs, RLCs, RLCp</i>
<i>DCTopology</i>	<i>Node</i>
<i>DG</i>	<i>PVD1, ESD1, EV1, EV2</i>
<i>DGProtection</i>	<i>DGPRCT1, DGPRCTExt</i>
<i>DataSeries</i>	<i>TimeSeries</i>
<i>DynLoad</i>	<i>ZIP, FLoad</i>
<i>Exciter</i>	<i>EXDC2, IEEEEX1, ESDC1A, ESDC2A, EXST1, ESST3A, SEXS, IEEET1, EXAC1, EXAC2, EXAC4, ESS</i>
<i>FreqMeasurement</i>	<i>BusFreq, BusROCOF</i>
<i>Information</i>	<i>Summary</i>
<i>Interface</i>	<i>Fortescue</i>
<i>Motor</i>	<i>Motor3, Motor5</i>
<i>OutputSelect</i>	<i>Output</i>
<i>PLL</i>	<i>PLL1, PLL2</i>
<i>PSS</i>	<i>IEEEEST, ST2CUT</i>
<i>PhasorMeasurement</i>	<i>PMU</i>
<i>RenAerodynamics</i>	<i>WTARA1, WTARV1</i>
<i>RenExciter</i>	<i>REECA1, REECA1E, REECA1G</i>
<i>RenGen</i>	<i>REGCA1, REGCPI, REGCV1, REGCV2, REGF1, REGF2, REGF3</i>
<i>RenGovernor</i>	<i>WTDTA1, WTDS</i>
<i>RenPitch</i>	<i>WTPTA1</i>
<i>RenPlant</i>	<i>REPCA1</i>
<i>RenTorque</i>	<i>WTTQA1</i>
<i>StaticACDC</i>	<i>VSCShunt</i>
<i>StaticGen</i>	<i>PV, Slack</i>

Table 1 – continued from previous page

Group	Models
<i>StaticLoad</i>	<i>PQ</i>
<i>StaticShunt</i>	<i>Shunt, ShuntTD, ShuntSw</i>
<i>SynGen</i>	<i>GENCLS, GENROU, PLBFU1</i>
<i>TimedEvent</i>	<i>Toggle, Fault, Alter</i>
<i>TurbineGov</i>	<i>TG2, TGOV1, TGOV1DB, TGOVIN, TGOVINDB, IEEEG1, IEESGO, GAST, HYGOV, HYGOVDB, H</i>
<i>VoltComp</i>	<i>IEEEVC</i>

## 5.1 ACLine

Common Parameters: u, name, bus1, bus2, r, x

Common Variables: v1, v2, a1, a2

Available models: *Line*

### 5.1.1 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	<i>bool</i>	
name		device name			
bus1		idx of from bus			
bus2		idx of to bus			
Sn	$S_n$	Power rating	100	<i>MW</i>	non_zero
fn	$f$	rated frequency	60	<i>Hz</i>	
Vn1	$V_{n1}$	AC voltage rating	110	<i>kV</i>	non_zero
Vn2	$V_{n2}$	rated voltage of bus2	110	<i>kV</i>	non_zero
r	$r$	line resistance	0.000	<i>p.u.</i>	<i>z</i>
x	$x$	line reactance	0.000	<i>p.u.</i>	non_zero,z
b		shared shunt susceptance	0	<i>p.u.</i>	<i>y</i>
g		shared shunt conductance	0	<i>p.u.</i>	<i>y</i>
b1	$b_1$	from-side susceptance	0	<i>p.u.</i>	<i>y</i>
g1	$g_1$	from-side conductance	0	<i>p.u.</i>	<i>y</i>
b2	$b_2$	to-side susceptance	0	<i>p.u.</i>	<i>y</i>
g2	$g_2$	to-side conductance	0	<i>p.u.</i>	<i>y</i>
trans		transformer branch flag	0	<i>bool</i>	
tap	$t_{ap}$	transformer branch tap ratio	1	<i>float</i>	non_negative
phi	$\phi$	transformer branch phase shift in rad	0	<i>radian</i>	
rate_a	$R_{ATEA}$	long-term flow limit (placeholder)	0	<i>MVA</i>	
rate_b	$R_{ATEB}$	short-term flow limit (placeholder)	0	<i>MVA</i>	
rate_c	$R_{ATEC}$	emergency flow limit (placeholder)	0	<i>MVA</i>	
owner		owner code			
xcoord		x coordinates			
ycoord		y coordinates			

## 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	$u \left( -1/t_{ap} v_1 v_2 (-b_{hk} \sin(\phi - a_1 + a_2) + g_{hk} \cos(\phi - a_1 + a_2)) + 1/t_{ap}^2 v_1^2 (g_h + g_{hk}) \right)$
a2	$a_2$	ExtAlgeb	$u \left( -1/t_{ap} v_1 v_2 (b_{hk} \sin(\phi - a_1 + a_2) + g_{hk} \cos(\phi - a_1 + a_2)) + v_2^2 (g_h + g_{hk}) \right)$
v1	$v_1$	ExtAlgeb	$u \left( -1/t_{ap} v_1 v_2 (-b_{hk} \cos(\phi - a_1 + a_2) - g_{hk} \sin(\phi - a_1 + a_2)) - 1/t_{ap}^2 v_1^2 (b_h + b_{hk}) \right)$
v2	$v_2$	ExtAlgeb	$u \left( 1/t_{ap} v_1 v_2 (b_{hk} \cos(\phi - a_1 + a_2) - g_{hk} \sin(\phi - a_1 + a_2)) - v_2^2 (b_h + b_{hk}) \right)$

## 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	$y_h$	$u (ib_h + g_h)$	ConstService
yk	$y_k$	$u (ib_k + g_k)$	ConstService
yhk	$y_{hk}$	$\frac{u}{r+i(x+1.0 \cdot 10^{-8})+1.0 \cdot 10^{-8}}$	ConstService
ghk	$g_{hk}$	$\text{re}(y_{hk})$	ConstService
bhk	$b_{hk}$	$\text{im}(y_{hk})$	ConstService
itap	$1/t_{ap}$	$\frac{1}{t_{ap}}$	ConstService
itap2	$1/t_{ap}^2$	$\frac{1}{t_{ap}^2}$	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)

## 5.2 ACShort

Common Parameters: u, name, bus1, bus2

Common Variables: v1, v2, a1, a2

Available models: *Jumper*

### 5.2.1 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	<i>u</i>	connection status	1	<i>bool</i>	
name		device name			
bus1		idx of from bus			
bus2		idx of to bus			

## 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	$P(1 - u) + u(a_1 - a_2)$
q	$Q$	Algeb	$P(1 - u) + u(v_1 - v_2)$
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 ACTopology

Common Parameters: u, name

Common Variables: a, v

Available models: [Bus](#)

### 5.3.1 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	<i>bool</i>	
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	$\theta_0$	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			

#### Variables

Name	Symbol	Type	Description	Unit	Properties
a	$\theta$	Algeb	voltage angle	rad	v_str
v	$V$	Algeb	voltage magnitude	p.u.	v_str

## Initialization Equations

Name	Symbol	Type	Initial Value
a	$\theta$	Algeb	$\theta_0 \cdot (1 - z_{flat}) + 1.0 \cdot 10^{-8}z_{flat}$
v	$V$	Algeb	$V_0 \cdot (1 - z_{flat}) + z_{flat}$

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
a	$\theta$	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)

## 5.4 Calculation

Group of classes that calculates based on other models.

Common Parameters: u, name

Available models: [ACE](#), [ACEc](#), [COI](#)

### 5.4.1 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`.

## Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	$u$	connection status	1	<i>bool</i>	
name		device name			
bus		bus idx for freq. measurement			mandatory
bias	$\beta$	bias parameter	1	<i>MW/0.1Hz</i>	power
busf		Optional BusFreq device idx			
area			0		

## Variables

Name	Symbol	Type	Description	Unit	Properties
ace	$ace$	Algeb	area control error	<i>p.u. (MW)</i>	
f	$f$	ExtAlgeb	Bus frequency	<i>p.u. (Hz)</i>	

## 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	$10 \cdot 1/S_{b,sys} \beta f_{sys} (v^{f_s} - 1) - ace$
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	

### 5.4.2 ACEc

Area Control Error model.

Continuous frequency sampling. System base frequency from `system.config.freq` is used.

Note: area idx is automatically retrieved from `bus`.

## 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	$\beta$	bias parameter	1	MW/0.1Hz	power
busf		Optional BusFreq device idx			
area			0		

## Variables

Name	Symbol	Type	Description	Unit	Properties
ace	<i>ace</i>	Algeb	area control error	<i>p.u. (MW)</i>	
f	<i>f</i>	ExtAlgeb	Bus frequency	<i>p.u. (Hz)</i>	

## Initialization Equations

Name	Symbol	Type	Initial Value
ace	<i>ace</i>	Algeb	
f	<i>f</i>	ExtAlgeb	

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
ace	<i>ace</i>	Algeb	$10 \cdot 1/S_{b,sys} \beta f_{sys} (f - 1) - ace$
f	<i>f</i>	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',)

### 5.4.3 COI

Center of inertia calculation class.

#### Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	$u$	connection status	1	<i>bool</i>	
name		device name			
M		Linearly stored SynGen.M	0		

#### Variables

Name	Symbol	Type	Description	Unit	Properties
wgen	$\omega_{gen}$	ExtState	Linearly stored SynGen.omega		
agen	$\delta_{gen}$	ExtState	Linearly stored SynGen.delta		
omega	$\omega_{coi}$	Algeb	COI speed		v_str,v_setter
delta	$\delta_{coi}$	Algeb	COI rotor angle		v_str,v_setter
omega_sub	$\omega_{sub}$	ExtAlgeb	COI frequency contribution of each generator		
delta_sub	$\delta_{sub}$	ExtAlgeb	COI angle contribution of each generator		

#### Initialization Equations

Name	Symbol	Type	Initial Value
wgen	$\omega_{gen}$	ExtState	
agen	$\delta_{gen}$	ExtState	
omega	$\omega_{coi}$	Algeb	$\omega_{gen,0,avg}$
delta	$\delta_{coi}$	Algeb	$\delta_{gen,0,avg}$
omega_sub	$\omega_{sub}$	ExtAlgeb	
delta_sub	$\delta_{sub}$	ExtAlgeb	

## Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
wgen	$\omega_{gen}$	ExtState	0	
agen	$\delta_{gen}$	ExtState	0	

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
omega	$\omega_{coi}$	Algeb	$-\omega_{coi}$
delta	$\delta_{coi}$	Algeb	$-\delta_{coi}$
omega_sub	$\omega_{sub}$	ExtAlgeb	$M_w \omega_{gen}$
delta_sub	$\delta_{sub}$	ExtAlgeb	$M_w \delta_{gen}$

## Services

Name	Symbol	Equation	Type
Mw	$M_w$	$\frac{M}{M_{tr}}$	ConstService
d0w	$\delta_{gen,0,w}$	$M_w \delta_{gen,0}$	ConstService
a0w	$\omega_{gen,0,w}$	$M_w \omega_{gen,0}$	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)

## 5.5 Collection

Collection of topology models

Common Parameters: u, name

Available models: [Area](#)

## 5.5.1 Area

Area model.

Area collects back references from the Bus model and the ACTopology group.

### Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	<i>u</i>	connection status	1	<i>bool</i>	
name		device name			

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)

## 5.6 DCLink

Basic DC links

Common Parameters: u, name

Available models: *Ground*, *R*, *L*, *C*, *RCp*, *RCs*, *RLs*, *RLCs*, *RLCp*

### 5.6.1 Ground

Ground model that sets the voltage of the connected DC node.

### Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	<i>u</i>	connection status	1	<i>bool</i>	
name		device name			
node		Node index			mandatory
voltage	<i>V</i> <sub>0</sub>	Ground voltage (typically 0)	0	<i>p.u.</i>	

## Variables

Name	Symbol	Type	Description	Unit	Properties
Idc	$I_{dc}$	Algeb	Fictitious current injection from ground		v_str
v	$v$	ExtAlgeb			

## Initialization Equations

Name	Symbol	Type	Initial Value
Idc	$I_{dc}$	Algeb	0
v	$v$	ExtAlgeb	

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
Idc	$I_{dc}$	Algeb	$u(-V_0 + v)$
v	$v$	ExtAlgeb	$-I_{dc}$

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)

## 5.6.2 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
R		DC line resistance	0.010	p.u.	non_zero,r

## Variables

Name	Symbol	Type	Description	Unit	Properties
Idc	$I_{dc}$	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	$I_{dc}$	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	$I_{dc}$	Algeb	$-I_{dc} + \frac{u(-v_1+v_2)}{R}$
v1	$v_1$	ExtAlgeb	$-I_{dc}$
v2	$v_2$	ExtAlgeb	$I_{dc}$

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)

### 5.6.3 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
L		DC line inductance	0.001	p.u.	non_zero,r

#### Variables

Name	Symbol	Type	Description	Unit	Properties
IL	$I_L$	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	$I_L$	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	$I_L$	State	$-u(v_1 - v_2)$	

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
v1	$v_1$	ExtAlgeb	$-I_L$
v2	$v_2$	ExtAlgeb	$I_L$

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)

## 5.6.4 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
C		DC capacitance	0.001	p.u.	non_zero,g

## Variables

Name	Symbol	Type	Description	Unit	Properties
vC	$v_C$	State	Capacitor current	p.u.	v_str
Idc	$I_{dc}$	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	$v_C$	State	0
Idc	$I_{dc}$	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	$v_C$	State	$-I_{dc}u$	

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
Idc	$I_{dc}$	Algeb	$I_{dc}(1 - u) + u(-v_1 + v_2 + v_C)$
v1	$v_1$	ExtAlgeb	$-I_{dc}$
v2	$v_2$	ExtAlgeb	$I_{dc}$

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)

## 5.6.5 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
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	$v_C$	State	Capacitor current	p.u.	v_str
Idc	$I_{dc}$	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	$v_C$	State	$v_1 - v_2$
Idc	$I_{dc}$	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	$v_C$	State	$-u (I_{dc} - \frac{v_C}{R})$	$C$

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
Idc	$I_{dc}$	Algeb	$I_{dc} (1 - u) + u (-v_1 + v_2 + v_C)$
v1	$v_1$	ExtAlgeb	$-I_{dc}$
v2	$v_2$	ExtAlgeb	$I_{dc}$

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)

## 5.6.6 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
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	$v_C$	State	Capacitor current	p.u.	v_str
Idc	$I_{dc}$	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	$v_C$	State	$v_1 - v_2$
Idc	$I_{dc}$	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	$v_C$	State	$-I_{dc}u$	$C$

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
Idc	$I_{dc}$	Algeb	$I_{dc}(1 - u) + u(-I_{dc}R - v_1 + v_2 + v_C)$
v1	$v_1$	ExtAlgeb	$-I_{dc}$
v2	$v_2$	ExtAlgeb	$I_{dc}$

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)

## 5.6.7 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
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	$I_L$	State	Inductance current	p.u.	v_str
Idc	$I_{dc}$	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	$I_L$	State	$\frac{v_1 - v_2}{R}$
Idc	$I_{dc}$	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	$I_L$	State	$u (-I_L R + v_1 - v_2)$	$L$

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
Idc	$I_{dc}$	Algeb	$-I_L u - I_{dc}$
v1	$v_1$	ExtAlgeb	$-I_{dc}$
v2	$v_2$	ExtAlgeb	$I_{dc}$

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)

## 5.6.8 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
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	$I_L$	State	Inductance current	p.u.	v_str
vC	$v_C$	State	Capacitor current	p.u.	v_str
Idc	$I_{dc}$	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	$I_L$	State	0
vC	$v_C$	State	$v_1 - v_2$
Idc	$I_{dc}$	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	$I_L$	State	$u (-I_L R + v_1 - v_2 - v_C)$	$L$
vC	$v_C$	State	$I_L u$	$C$

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
Idc	$I_{dc}$	Algeb	$-I_L - I_{dc}$
v1	$v_1$	ExtAlgeb	$-I_{dc}$
v2	$v_2$	ExtAlgeb	$I_{dc}$

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)

## 5.6.9 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
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	$I_L$	State	Inductance current	p.u.	v_str
vC	$v_C$	State	Capacitor current	p.u.	v_str
Idc	$I_{dc}$	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	$I_L$	State	0
vC	$v_C$	State	$v_1 - v_2$
Idc	$I_{dc}$	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	$I_L$	State	$uv_C$	$L$
vC	$v_C$	State	$-u \left( -I_L + I_{dc} - \frac{v_C}{R} \right)$	$C$

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
Idc	$I_{dc}$	Algeb	$I_{dc} (1 - u) + u (-v_1 + v_2 + v_C)$
v1	$v_1$	ExtAlgeb	$-I_{dc}$
v2	$v_2$	ExtAlgeb	$I_{dc}$

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)

## 5.7 DCTopology

Common Parameters: u, name

Common Variables: v

Available models: *Node*

### 5.7.1 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	<i>bool</i>	
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	<i>p.u.</i>	
xcoord		x coordinate (longitude)	0		
ycoord		y coordinate (latitude)	0		
area		Area code			
zone		Zone code			
owner		Owner code			

## Variables

Name	Symbol	Type	Description	Unit	Properties
v	$V_{dc}$	Algeb	voltage magnitude	<i>p.u.</i>	v_str

## Initialization Equations

Name	Symbol	Type	Initial Value
v	$V_{dc}$	Algeb	$V_{dc0} \cdot (1 - z_{flat}) + z_{flat}$

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
v	$V_{dc}$	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)

## 5.8 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*, *EVI*, *EV2*

### 5.8.1 PVD1

WECC Distributed PV model.

Device power rating is specified in  $Sn$ . Output currents are named  $I_{pout\_y}$  and  $I_{qout\_y}$ . Output power can be computed as  $P_e = I_{pout\_y} * v$  and  $Q_e = I_{qout\_y} * v$ .

Frequency tripping response points  $ft0$ ,  $ft1$ ,  $ft2$ , and  $ft3$  must be monotonically 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.

Modifications to the active and reactive power references, typically by an external scheduling program, should write to  $pref0.v$  and  $qref0.v$  in place. AGC signals should write to  $pext0.v$  in place.

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

continues on

Table 2 – continued from previous page

Name	Symbol	Description	Default	Unit	Properties
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
dqv	$dq/dv$	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,power
ialim	$I_{alim}$	Apparent power limit	1.300		non_zero,non_neg
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	$\gamma_p$	Ratio of PVD1.pref0 w.r.t to that of static PV	1		
gammaq	$\gamma_q$	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		

## Variables

Name	Symbol	Type	Description	Unit	Properties
Ipout_y	$y_{Ipout}$	State	State in lag transfer function		v_str
Iqout_y	$y_{Iqout}$	State	State in lag transfer function		v_str
fHz	$f_{Hz}$	Algeb	frequency in Hz	Hz	v_str
Ffl	$F_{fl}$	Algeb	Coeff. for under frequency		v_str
Ffh	$F_{fh}$	Algeb	Coeff. for over frequency		v_str
Fdev	$f_{dev}$	Algeb	Frequency deviation	Hz	v_str
DB_y	$y_{DB}$	Algeb	Deadband type 1 output		v_str
Fvl	$F_{vl}$	Algeb	Coeff. for under voltage		v_str
Fvh	$F_{vh}$	Algeb	Coeff. for over voltage		v_str
vp	$V_p$	Algeb	Sensed positive voltage		v_str
Pext	$P_{ext}$	Algeb	External power signal (for AGC)		v_str
Pref	$P_{ref}$	Algeb	Reference power signal (for scheduling setpoint)		v_str
Psum	$P_{tot}$	Algeb	Sum of P signals		v_str
Qdrp	$Q_{drp}$	Algeb	External power signal (for AGC)		v_str
Qref	$Q_{ref}$	Algeb	Reference power signal (for scheduling setpoint)		v_str
Qsum	$Q_{tot}$	Algeb	Sum of Q signals		v_str
Ipul	$I_{p,ul}$	Algeb	Ipcmd before Ip hard limit		v_str
Iqul	$I_{q,ul}$	Algeb	Iqcmd before Iq hard limit		v_str
Ipmax	$I_{pmax}$	Algeb	Upper limit of Ip		v_str
Iqmax	$I_{qmax}$	Algeb	Upper limit of Iq		v_str
Ipcmd_x	$x_{Ipcmd}$	Algeb	Value before limiter		v_str
Ipcmd_y	$y_{Ipcmd}$	Algeb	Output after limiter and post gain		v_str
Iqcmd_x	$x_{Iqcmd}$	Algeb	Value before limiter		v_str
Iqcmd_y	$y_{Iqcmd}$	Algeb	Output after limiter and post gain		v_str
a	$\theta$	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

Name	Sym-	Type	Initial Value
	bol		
Ipout_	$y_{Ipout}$	State	$1.0y_{Ipcmd}$
Iqout_	$y_{Iqout}$	State	$1.0y_{Iqcmd}$
fHz	$f_{Hz}$	Al- geb	$ff_n$
Ffl	$F_{fl}$	Al- geb	$K_{ft01}z_i^{FL1} (f_{Hz} - f_{t0}) + z_u^{FL1}$
Ffh	$F_{fh}$	Al- geb	$z_i^{FL2} (K_{ft23} (-f_{Hz} + f_{t2}) + 1) + z_l^{FL2}$
Fdev	$f_{dev}$	Al- geb	$f_n - f_{Hz}$
DB_y	$y_{DB}$	Al- geb	$D_{dn} \left( f_{dev}z_u^{db_{DB}} + z_l^{db_{DB}} (-f_{dbd} + f_{dev}) \right)$
Fvl	$F_{vl}$	Al- geb	$K_{vt01}z_i^{VL1} (V - V_{t0}) + z_u^{VL1}$
Fvh	$F_{vh}$	Al- geb	$z_i^{VL2} (K_{vt23} (-V + V_{t2}) + 1) + z_l^{VL2}$
vp	$V_p$	Al- geb	$Vz_i^{VL0} + 0.01z_l^{VL0}$
Pext	$P_{ext}$	Al- geb	$P_{ext0}u$
Pref	$P_{ref}$	Al- geb	$P_{ref0}u$
Psum	$P_{tot}$	Al- geb	$u(P_{ext} + P_{ref} + y_{DB})$
Qdrp	$Q_{drp}$	Al- geb	$dq/dvuz_i^{VQ2} (-V_{comp} + v_1) + q_{mn}z_u^{VQ2} + q_{mx}uz_l^{VQ1} + uz_i^{VQ1} (dq/dv (-V_{comp} + V_{qu}) + q_{mx})$
Qref	$Q_{ref}$	Al- geb	$Q_{ref0}u$
Qsum	$Q_{tot}$	Al- geb	$u(Q_{ref0} + dq/dvuz_i^{VQ2} (-V_{comp} + v_1) + q_{mn}z_u^{VQ2} + q_{mx}uz_l^{VQ1} + uz_i^{VQ1} (dq/dv (-V_{comp} + V_{qu}) + q_{mx}))$
Ipul	$I_{p,ul}$	Al- geb	$\frac{P_{tot}z_i^{PHL} + p_{mx}z_u^{PHL}}{V_p}$
Iqul	$I_{q,ul}$	Al- geb	$\frac{Q_{tot}}{V_p}$
Ip- max	$I_{pmax}$	Al- geb	$I_{alim}s_1^{SW_{PQ}} + \sqrt{I_{pmax0}^2}s_0^{SW_{PQ}}$
Iq- max	$I_{qmax}$	Al- geb	$I_{alim}s_0^{SW_{PQ}} + \sqrt{I_{qmax0}^2}s_1^{SW_{PQ}}$
Ipcmd	$x_{Ipcmd}$	Al- geb	$I_{p,ul}$
Ipcmd	$y_{Ipcmd}$	Al- geb	$I_{pmax}z_u^{\lim_{Ipcmd}} (F_{fh}F_{fl}F_{vh}F_{vl}z_{rec} - z_{rec} + 1) + x_{Ipcmd}z_i^{\lim_{Ipcmd}} (F_{fh}F_{fl}F_{vh}F_{vl}z_{rec} - z_{rec} + 1)$
464	$x_{Iqcmd}$	Al- geb	$I_{q,ul}$
Iqcmd	$y_{Iqcmd}$	Al- geb	$-I_{qmax}z_l^{\lim_{Iqcmd}} (F_{fh}F_{fl}F_{vh}F_{vl}z_{rec} - z_{rec} + 1) + x_{Iqcmd}z_i^{\lim_{Iqcmd}} (F_{fh}F_{fl}F_{vh}F_{vl}z_{rec} - z_{rec} + 1)$

## Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
Ipout_y	$y_{Ipout}$	State	$1.0y_{Ipcmd} - y_{Ipout}$	$T_{ip}$
Iqout_y	$y_{Iqout}$	State	$1.0y_{Iqcmd} - y_{Iqout}$	$T_{iq}$

## Algebraic Equations

Name	Sym-	Type	RHS of Equation "0 = g(x, y)"
	bol		
fHz	$f_{Hz}$	Al- geb	$ff_n - f_{Hz}$
Ffl	$F_{fl}$	Al- geb	$-F_{fl} + K_{ft01}z_i^{FL1} (f_{Hz} - f_{t0}) + z_u^{FL1}$
Ffh	$F_{fh}$	Al- geb	$-F_{fh} + z_i^{FL2} (K_{ft23} (-f_{Hz} + f_{t2}) + 1) + z_l^{FL2}$
Fdev	$f_{dev}$	Al- geb	$f_n - f_{Hz} - f_{dev}$
DB_y	$y_{DB}$	Al- geb	$D_{dn} \left( f_{dev} z_u^{db_{DB}} + z_l^{db_{DB}} (-f_{dbd} + f_{dev}) \right) - y_{DB}$
Fvl	$F_{vl}$	Al- geb	$-F_{vl} + K_{vt01}z_i^{VL1} (V - V_{t0}) + z_u^{VL1}$
Fvh	$F_{vh}$	Al- geb	$-F_{vh} + z_i^{VL2} (K_{vt23} (-V + V_{t2}) + 1) + z_l^{VL2}$
vp	$V_p$	Al- geb	$V z_i^{VLo} - V_p + 0.01 z_l^{VLo}$
Pext	$P_{ext}$	Al- geb	$P_{ext0}u - P_{ext}$
Pref	$P_{ref}$	Al- geb	$P_{ref0}u - P_{ref}$
Psum	$P_{tot}$	Al- geb	$-P_{tot} + u (P_{ext} + P_{ref} + y_{DB})$
Qdrp	$Q_{drp}$	Al- geb	$-Q_{drp} + dq/dvuz_i^{VQ2} (-V_{comp} + v_1) + q_{mn}z_u^{VQ2} + q_{mx}uz_l^{VQ1} + uz_i^{VQ1} (dq/dv (-V_{comp} + V_{qu}) + q_{mx})$
Qref	$Q_{ref}$	Al- geb	$Q_{ref0}u - Q_{ref}$
Qsum	$Q_{tot}$	Al- geb	$-Q_{tot} + u (Q_{drp} + Q_{ref})$
Ipul	$I_{p,ul}$	Al- geb	$-I_{p,ul} + \frac{P_{tot}z_i^{PHL} + p_{mx}z_u^{PHL}}{V_p}$
Iqul	$I_{q,ul}$	Al- geb	$-I_{q,ul} + \frac{Q_{tot}}{V_p}$
Ip- max	$I_{pmax}$	Al- geb	$I_{alim}s_1^{SW_{PQ}} - I_{pmax} + \sqrt{I_{pmax}^2}s_0^{SW_{PQ}}$
Iq- max	$I_{qmax}$	Al- geb	$I_{alim}s_0^{SW_{PQ}} - I_{qmax} + \sqrt{I_{qmax}^2}s_1^{SW_{PQ}}$
Ipcmc	$x_{Ipcm}$	Al- geb	$I_{p,ul} - x_{Ipcmd}$
Ipcmc	$y_{Ipcm}$	Al- geb	$I_{pmax} \lim_{Ipcmd} (F_{fh} F_{fl} F_{vh} F_{vl} z_{rec} - z_{rec} + 1)$
Iqcmci	$x_{Iqcm}$	Al- geb	$x_{Ipcmd} \lim_{Ipcmd} (F_{fh} F_{fl} F_{vh} F_{vl} z_{rec} - z_{rec} + 1) - y_{Ipcmd}$
Iqcmci	$y_{Iqcm}$	Al- geb	$I_{q,ul} - x_{Iqcmd}$
Iqcmci	$y_{Iqcm}$	Al- geb	$-I_{qmax} \lim_{Iqcmd} (F_{fh} F_{fl} F_{vh} F_{vl} z_{rec} - z_{rec} + 1)$
Iqcmci	$y_{Iqcm}$	Al- geb	$I_{qmax} \lim_{Iqcmd} (F_{fh} F_{fl} F_{vh} F_{vl} z_{rec} - z_{rec} + 1)$
Iqcmci	$y_{Iqcm}$	Al- geb	$x_{Iqcmd} \lim_{Iqcmd} (F_{fh} F_{fl} F_{vh} F_{vl} z_{rec} - z_{rec} + 1) - y_{Iqcmd}$

## Services

Name	Symbol	Equation	Type
pref0	$P_{ref0}$	$P_{0s}\gamma_p$	ConstService
qref0	$Q_{ref0}$	$Q_{0s}\gamma_q$	ConstService
Kft01	$K_{ft01}$	$\frac{1}{-f_{t0}+f_{t1}}$	ConstService
Kft23	$K_{ft23}$	$\frac{1}{-f_{t2}+f_{t3}}$	ConstService
Kvt01	$K_{vt01}$	$\frac{1}{-V_{t0}+V_{t1}}$	ConstService
Kvt23	$K_{vt23}$	$\frac{1}{-V_{t2}+V_{t3}}$	ConstService
Pext0	$P_{ext0}$	0	ConstService
Vcomp	$V_{comp}$	$ Ve^{i\theta} + ix_c(y_{Ipout} + iy_{Iqout}) $	VarService
Vqu	$V_{qu}$	$v_1 - \frac{Q_{ref0}-q_{mn}}{dq/dv}$	ConstService
Vql	$V_{ql}$	$v_0 + \frac{-Q_{ref0}+q_{mx}}{dq/dv}$	ConstService
Ipmаксq	$I_{pmax}^2$	FixPiecewise $\left( \left( 0, I_{alim}^2 - (y_{Iqcmd})^2 \leq 0 \right), \left( I_{alim}^2 - (y_{Iqcmd})^2, \text{True} \right) \right)$	VarService
Ip- maxsq0	$I_{pmax0}^2$	FixPiecewise $\left( \left( 0, I_{alim}^2 - \frac{Q_{ref0}^2 u^2}{V^2} \leq 0 \right), \left( I_{alim}^2 - \frac{Q_{ref0}^2 u^2}{V^2}, \text{True} \right) \right)$	ConstService
Iq максq	$I_{qmax}^2$	FixPiecewise $\left( \left( 0, I_{alim}^2 - (y_{Ipcmd})^2 \leq 0 \right), \left( I_{alim}^2 - (y_{Ipcmd})^2, \text{True} \right) \right)$	VarService
Iq- maxsq0	$I_{qmax0}^2$	FixPiecewise $\left( \left( 0, I_{alim}^2 - \frac{P_{ref0}^2 u^2}{V^2} \leq 0 \right), \left( I_{alim}^2 - \frac{P_{ref0}^2 u^2}{V^2}, \text{True} \right) \right)$	ConstService

## 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

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)
ad- just_upper		1	adjust upper limit	(0, 1)
plim	$P_{lim}$	1	enable input power limit check bound by [0, pmx]	(0, 1)

## 5.8.2 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](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
dqv	$dq/dv$	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,pow
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
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

continues on next page

Table 3 – continued from previous page

Name	Symbol	Description	Default	Unit	Properties
gammap	$\gamma_p$	Ratio of PVD1.pref0 w.r.t to that of static PV	1		
gammaq	$\gamma_q$	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		

## Variables

Name	Symbol	Type	Description	Unit	Properties
Ipout_y	$y_{Ipout}$	State	State in lag transfer function		v_str
Iqout_y	$y_{Iqout}$	State	State in lag transfer function		v_str
pIG_y	$y_{pIG}$	State	Integrator output		v_str
SOC	$SOC$	AliasState	Alias for state of charge		
fHz	$f_{Hz}$	Algeb	frequency in Hz	Hz	v_str
Ffl	$F_{fl}$	Algeb	Coeff. for under frequency		v_str
Ffh	$F_{fh}$	Algeb	Coeff. for over frequency		v_str
Fdev	$f_{dev}$	Algeb	Frequency deviation	Hz	v_str
DB_y	$y_{DB}$	Algeb	Deadband type 1 output		v_str
Fvl	$F_{vl}$	Algeb	Coeff. for under voltage		v_str
Fvh	$F_{vh}$	Algeb	Coeff. for over voltage		v_str
vp	$V_p$	Algeb	Sensed positive voltage		v_str
Pext	$P_{ext}$	Algeb	External power signal (for AGC)		v_str
Pref	$P_{ref}$	Algeb	Reference power signal (for scheduling setpoint)		v_str
Psum	$P_{tot}$	Algeb	Sum of P signals		v_str
Qdrp	$Q_{drp}$	Algeb	External power signal (for AGC)		v_str
Qref	$Q_{ref}$	Algeb	Reference power signal (for scheduling setpoint)		v_str
Qsum	$Q_{tot}$	Algeb	Sum of Q signals		v_str
Ipul	$I_{p,ul}$	Algeb	Ipcmd before Ip hard limit		v_str
Iqul	$I_{q,ul}$	Algeb	Iqcmd before Iq hard limit		v_str
Ipmax	$I_{pmax}$	Algeb	Upper limit of Ip		v_str
Iqmax	$I_{qmax}$	Algeb	Upper limit of Iq		v_str
Ipcmd_x	$x_{Ipcmd}$	Algeb	Value before limiter		v_str
Ipcmd_y	$y_{Ipcmd}$	Algeb	Output after limiter and post gain		v_str
Iqcmd_x	$x_{Iqcmd}$	Algeb	Value before limiter		v_str
Iqcmd_y	$y_{Iqcmd}$	Algeb	Output after limiter and post gain		v_str
Ipmin	$I_{pmin}$	Algeb	Minimum value of Ip		v_str

continues on next page

Table 4 – continued from previous page

Name	Symbol	Type	Description	Unit	Properties
a	$\theta$	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

Name	Symbol	Type	Initial Value
Ipout_y	$y_{Ipout}$	State	$1.0y_{Ipcmd}$
Iqout_y	$y_{Iqout}$	State	$1.0y_{Iqcmd}$
pIG_y	$y_{pIG}$	State	$SOC_{init}$
SOC	$SOC$	AliasState	
fHz	$f_{Hz}$	Algeb	$f f_n$
Ffl	$F_{fl}$	Algeb	$K_{ft01}z_i^{FL1} (f_{Hz} - f_{t0}) + z_u^{FL1}$
Ffh	$F_{fh}$	Algeb	$z_i^{FL2} (K_{ft23} (-f_{Hz} + f_{t2}) + 1) + z_l^{FL2}$
Fdev	$f_{dev}$	Algeb	$f_n - f_{Hz}$
DB_y	$y_{DB}$	Algeb	$D_{dn} \left( f_{dev}z_u^{db_{DB}} + z_l^{db_{DB}} (-f_{dbd} + f_{dev}) \right)$
Fvl	$F_{vl}$	Algeb	$K_{vt01}z_i^{VL1} (V - V_{t0}) + z_u^{VL1}$
Fvh	$F_{vh}$	Algeb	$z_i^{VL2} (K_{vt23} (-V + V_{t2}) + 1) + z_l^{VL2}$
vp	$V_p$	Algeb	$V z_i^{VLo} + 0.01 z_l^{VLo}$
Pext	$P_{ext}$	Algeb	$P_{ext0}u$
Pref	$P_{ref}$	Algeb	$P_{ref0}u$
Psum	$P_{tot}$	Algeb	$u (P_{ext} + P_{ref} + y_{DB})$
Qdrp	$Q_{drp}$	Algeb	$dq/dv u z_i^{VQ2} (-V_{comp} + v_1) + q_{mn} z_u^{VQ2} + q_{mx} u z_l^{VQ1} + u z_i^{VQ1} (dq/dv (-V_{comp} + V_1))$
Qref	$Q_{ref}$	Algeb	$Q_{ref0}u$
Qsum	$Q_{tot}$	Algeb	$u (Q_{ref0} + dq/dv u z_i^{VQ2} (-V_{comp} + v_1) + q_{mn} z_u^{VQ2} + q_{mx} u z_l^{VQ1} + u z_i^{VQ1} (dq/dv (-V_{comp} + V_1)))$
Ipul	$I_{p,ul}$	Algeb	$\frac{P_{tot} z_i^{PHL} + p_{mx} z_u^{PHL}}{V_p}$
Iqul	$I_{q,ul}$	Algeb	$\frac{Q_{tot}}{V_p}$
Ipmax	$I_{pmax}$	Algeb	$(1 - z_l^{SOClim}) \left( I_{alim} s_1^{SW_{PQ}} + \sqrt{I_{pmax0}^2 s_0^{SW_{PQ}}} \right)$
Iqmax	$I_{qmax}$	Algeb	$I_{alim} s_0^{SW_{PQ}} + \sqrt{I_{qmax0}^2 s_1^{SW_{PQ}}}$
Ipcmd_x	$x_{Ipcmd}$	Algeb	$I_{p,ul}$
Ipcmd_y	$y_{Ipcmd}$	Algeb	$I_{pmax} z_u^{lim_{Ipcmd}} (F_{fh} F_{fl} F_{vh} F_{vl} z_{rec} - z_{rec} + 1) + x_{Ipcmd} z_i^{lim_{Ipcmd}} (F_{fh} F_{fl} F_{vh} F_{vl} z_{rec})$
Iqcmd_x	$x_{Iqcmd}$	Algeb	$I_{q,ul}$
Iqcmd_y	$y_{Iqcmd}$	Algeb	$-I_{qmax} z_l^{lim_{Iqcmd}} (F_{fh} F_{fl} F_{vh} F_{vl} z_{rec} - z_{rec} + 1) + I_{qmax} z_u^{lim_{Iqcmd}} (F_{fh} F_{fl} F_{vh} F_{vl} z_{rec})$
Ipmin	$I_{pmin}$	Algeb	$(z_u^{SOClim} - 1) \left( I_{alim} s_1^{SW_{PQ}} + \sqrt{I_{pmax0}^2 s_0^{SW_{PQ}}} \right)$
a	$\theta$	ExtAlgeb	
v	$V$	ExtAlgeb	
f	$f$	ExtAlgeb	

## Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
Ipout_y	$y_{Ipout}$	State	$1.0y_{Ipcmd} - y_{Ipout}$	$T_{ip}$
Iqout_y	$y_{Iqout}$	State	$1.0y_{Iqcmd} - y_{Iqout}$	$T_{iq}$
pIG_y	$y_{pIG}$	State	$\frac{S_{b,sys} \left( -H_C V y_{Ipout} z_1^{LTN} - \frac{V y_{Ipout} z_0^{LTN}}{H_D} \right)}{3600 E_n}$	$T_f$
SOC	$SOC$	AliasState	0	

## Algebraic Equations

Name	Sym- bol	Type	RHS of Equation "0 = g(x, y)"
fHz	$f_{Hz}$	Al- geb	$ff_n - f_{Hz}$
Ffl	$F_{fl}$	Al- geb	$-F_{fl} + K_{ft01}z_i^{FL1} (f_{Hz} - f_{t0}) + z_u^{FL1}$
Ffh	$F_{fh}$	Al- geb	$-F_{fh} + z_i^{FL2} (K_{ft23} (-f_{Hz} + f_{t2}) + 1) + z_l^{FL2}$
Fdev	$f_{dev}$	Al- geb	$f_n - f_{Hz} - f_{dev}$
DB_y	$y_{DB}$	Al- geb	$D_{dn} \left( f_{dev} z_u^{db_{DB}} + z_l^{db_{DB}} (-f_{dbd} + f_{dev}) \right) - y_{DB}$
Fvl	$F_{vl}$	Al- geb	$-F_{vl} + K_{vt01}z_i^{VL1} (V - V_{t0}) + z_u^{VL1}$
Fvh	$F_{vh}$	Al- geb	$-F_{vh} + z_i^{VL2} (K_{vt23} (-V + V_{t2}) + 1) + z_l^{VL2}$
vp	$V_p$	Al- geb	$V z_i^{VLo} - V_p + 0.01 z_l^{VLo}$
Pext	$P_{ext}$	Al- geb	$P_{ext0}u - P_{ext}$
Pref	$P_{ref}$	Al- geb	$P_{ref0}u - P_{ref}$
Psum	$P_{tot}$	Al- geb	$-P_{tot} + u (P_{ext} + P_{ref} + y_{DB})$
Qdrp	$Q_{drp}$	Al- geb	$-Q_{drp} + dq/dvuz_i^{VQ2} (-V_{comp} + v_1) + q_{mn}z_u^{VQ2} + q_{mx}uz_l^{VQ1} + uz_i^{VQ1} (dq/dv (-V_{comp} + V_{qu}) + q_{mx})$
Qref	$Q_{ref}$	Al- geb	$Q_{ref0}u - Q_{ref}$
Qsum	$Q_{tot}$	Al- geb	$-Q_{tot} + u (Q_{drp} + Q_{ref})$
Ipul	$I_{p,ul}$	Al- geb	$-I_{p,ul} + \frac{P_{tot}z_i^{PHL} + p_{mx}z_u^{PHL}}{V_p}$
Iqul	$I_{q,ul}$	Al- geb	$-I_{q,ul} + \frac{Q_{tot}}{V_p}$
Ip- max	$I_{pmax}$	Al- geb	$-I_{pmax} + (1 - z_l^{SOClim}) \left( I_{alim} s_1^{SW_{PQ}} + \sqrt{I_{pmax}^2 s_0^{SW_{PQ}}} \right)$
Iq- max	$I_{qmax}$	Al- geb	$I_{alim} s_0^{SW_{PQ}} - I_{qmax} + \sqrt{I_{qmax}^2 s_1^{SW_{PQ}}}$
Ipcmc	$x_{Ipcm}$	Al- geb	$I_{p,ul} - x_{Ipcmd}$
Ipcmc	$y_{Ipcm}$	Al- geb	$I_{pmax} z_u^{\lim_{Ipcmd}} (F_{fh} F_{fl} F_{vh} F_{vl} z_{rec} - z_{rec} + 1) + x_{Ipcmd} z_i^{\lim_{Ipcmd}} (F_{fh} F_{fl} F_{vh} F_{vl} z_{rec} - z_{rec} + 1) - y_{Ipcmd}$
Iqcmci	$x_{Iqcm}$	Al- geb	$I_{q,ul} - x_{Iqcmd}$
58emDG	$y_{Iqcm}$	Al- geb	$-I_{qmax} z_l^{\lim_{Iqcmd}} (F_{fh} F_{fl} F_{vh} F_{vl} z_{rec} - z_{rec} + 1) + I_{qmax} z_u^{\lim_{Iqcmd}} (F_{fh} F_{fl} F_{vh} F_{vl} z_{rec} - z_{rec} + 1) + x_{Iqcmd} z_i^{\lim_{Iqcmd}} (F_{fh} F_{fl} F_{vh} F_{vl} z_{rec} - z_{rec} + 1) - y_{Iqcmd}$

## Services

Name	Symbol	Equation	Type
pref0	$P_{ref0}$	$P_{0s}\gamma_p$	ConstService
qref0	$Q_{ref0}$	$Q_{0s}\gamma_q$	ConstService
Kft01	$K_{ft01}$	$\frac{1}{-f_{t0}+f_{t1}}$	ConstService
Kft23	$K_{ft23}$	$\frac{1}{-f_{t2}+f_{t3}}$	ConstService
Kvt01	$K_{vt01}$	$\frac{1}{-V_{t0}+V_{t1}}$	ConstService
Kvt23	$K_{vt23}$	$\frac{1}{-V_{t2}+V_{t3}}$	ConstService
Pext0	$P_{ext0}$	0	ConstService
Vcomp	$V_{comp}$	$ Ve^{i\theta} + ix_c(y_{Ipout} + iy_{Iqout}) $	VarService
Vqu	$V_{qu}$	$v_1 - \frac{Q_{ref0}-q_{mn}}{dq/dv}$	ConstService
Vql	$V_{ql}$	$v_0 + \frac{-Q_{ref0}+q_{mx}}{dq/dv}$	ConstService
Ipmаксsq	$I_{pmax}^2$	FixPiecewise $\left( \left( 0, I_{alim}^2 - (y_{Ipcmd})^2 \leq 0 \right), \left( I_{alim}^2 - (y_{Ipcmd})^2, \text{True} \right) \right)$	VarService
Ip- maxsq0	$I_{pmax0}^2$	FixPiecewise $\left( \left( 0, I_{alim}^2 - \frac{Q_{ref0}^2 u^2}{V^2} \leq 0 \right), \left( I_{alim}^2 - \frac{Q_{ref0}^2 u^2}{V^2}, \text{True} \right) \right)$	ConstService
Iq максsq	$I_{qmax}^2$	FixPiecewise $\left( \left( 0, I_{alim}^2 - (y_{Ipcmd})^2 \leq 0 \right), \left( I_{alim}^2 - (y_{Ipcmd})^2, \text{True} \right) \right)$	VarService
Iq- maxsq0	$I_{qmax0}^2$	FixPiecewise $\left( \left( 0, I_{alim}^2 - \frac{P_{ref0}^2 u^2}{V^2} \leq 0 \right), \left( I_{alim}^2 - \frac{P_{ref0}^2 u^2}{V^2}, \text{True} \right) \right)$	ConstService

## 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	$Ipcmd$	GainLimiter	Ip with limiter and coeff.
Iqcmd	$Iqcmd$	GainLimiter	Iq with limiter and coeff.
Ipout	$Ipout$	Lag	Output Ip filter
Iqout	$Iqout$	Lag	Output Iq filter
pIG	$pIG$	Integrator	State of charge

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)
ad-just_upper		1	adjust upper limit	(0, 1)
plim	$P_{lim}$	1	enable input power limit check bound by [0, pmx]	(0, 1)

### 5.8.3 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
dqv	$dq/dv$	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,pow
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
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	$\gamma_p$	Ratio of PVD1.pref0 w.r.t to that of static PV	1		

continues on next page

Table 6 – continued from previous page

Name	Symbol	Description	Default	Unit	Properties
gammaq	$\gamma_q$	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

## Variables

Name	Symbol	Type	Description	Unit	Properties
Ipout_y	$y_{Ipout}$	State	State in lag transfer function		v_str
Iqout_y	$y_{Iqout}$	State	State in lag transfer function		v_str
pIG_y	$y_{pIG}$	State	Integrator output		v_str
SOC	$SOC$	AliasState	Alias for state of charge		
fHz	$f_{Hz}$	Algeb	frequency in Hz	Hz	v_str
Ffl	$F_{fl}$	Algeb	Coeff. for under frequency		v_str
Ffh	$F_{fh}$	Algeb	Coeff. for over frequency		v_str
Fdev	$f_{dev}$	Algeb	Frequency deviation	Hz	v_str
DB_y	$y_{DB}$	Algeb	Deadband type 1 output		v_str
Fvl	$F_{vl}$	Algeb	Coeff. for under voltage		v_str
Fvh	$F_{vh}$	Algeb	Coeff. for over voltage		v_str
vp	$V_p$	Algeb	Sensed positive voltage		v_str
Pext	$P_{ext}$	Algeb	External power signal (for AGC)		v_str
Pref	$P_{ref}$	Algeb	Reference power signal (for scheduling setpoint)		v_str
Psum	$P_{tot}$	Algeb	Sum of P signals		v_str
Qdrp	$Q_{drp}$	Algeb	External power signal (for AGC)		v_str
Qref	$Q_{ref}$	Algeb	Reference power signal (for scheduling setpoint)		v_str
Qsum	$Q_{tot}$	Algeb	Sum of Q signals		v_str
Ipul	$I_{p,ul}$	Algeb	Ipcmd before Ip hard limit		v_str
Iqul	$I_{q,ul}$	Algeb	Iqcmd before Iq hard limit		v_str
Ipmax	$I_{pmax}$	Algeb	Upper limit of Ip		v_str
Iqmax	$I_{qmax}$	Algeb	Upper limit of Iq		v_str
Ipcmd_x	$x_{Ipcmd}$	Algeb	Value before limiter		v_str
Ipcmd_y	$y_{Ipcmd}$	Algeb	Output after limiter and post gain		v_str
Iqcmd_x	$x_{Iqcmd}$	Algeb	Value before limiter		v_str
Iqcmd_y	$y_{Iqcmd}$	Algeb	Output after limiter and post gain		v_str
Ipmin	$I_{pmin}$	Algeb	Minimum value of Ip		v_str

continues on next page

Table 7 – continued from previous page

Name	Symbol	Type	Description	Unit	Properties
a	$\theta$	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

Name	Symbol	Type	Initial Value
Ipout_y	$y_{Ipout}$	State	$1.0y_{Ipcmd}$
Iqout_y	$y_{Iqout}$	State	$1.0y_{Iqcmd}$
pIG_y	$y_{pIG}$	State	$SOC_{init}$
SOC	$SOC$	AliasState	
fHz	$f_{Hz}$	Algeb	$f f_n$
Ffl	$F_{fl}$	Algeb	$K_{ft01}z_i^{FL1} (f_{Hz} - f_{t0}) + z_u^{FL1}$
Ffh	$F_{fh}$	Algeb	$z_i^{FL2} (K_{ft23} (-f_{Hz} + f_{t2}) + 1) + z_l^{FL2}$
Fdev	$f_{dev}$	Algeb	$f_n - f_{Hz}$
DB_y	$y_{DB}$	Algeb	$D_{dn} \left( f_{dev}z_u^{db_{DB}} + z_l^{db_{DB}} (-f_{dbd} + f_{dev}) \right)$
Fvl	$F_{vl}$	Algeb	$K_{vt01}z_i^{VL1} (V - V_{t0}) + z_u^{VL1}$
Fvh	$F_{vh}$	Algeb	$z_i^{VL2} (K_{vt23} (-V + V_{t2}) + 1) + z_l^{VL2}$
vp	$V_p$	Algeb	$V z_i^{VLo} + 0.01 z_l^{VLo}$
Pext	$P_{ext}$	Algeb	$P_{ext0}u$
Pref	$P_{ref}$	Algeb	$P_{ref0}u$
Psum	$P_{tot}$	Algeb	$u (P_{ext} + P_{ref} + y_{DB})$
Qdrp	$Q_{drp}$	Algeb	$dq/dv u z_i^{VQ2} (-V_{comp} + v_1) + q_{mn} z_u^{VQ2} + q_{mx} u z_l^{VQ1} + u z_i^{VQ1} (dq/dv (-V_{comp} + V_1))$
Qref	$Q_{ref}$	Algeb	$Q_{ref0}u$
Qsum	$Q_{tot}$	Algeb	$u (Q_{ref0} + dq/dv u z_i^{VQ2} (-V_{comp} + v_1) + q_{mn} z_u^{VQ2} + q_{mx} u z_l^{VQ1} + u z_i^{VQ1} (dq/dv (-V_{comp} + V_1)))$
Ipul	$I_{p,ul}$	Algeb	$\frac{P_{tot} z_i^{PHL} + p_{mn} z_l^{PHL} + p_{mx} z_u^{PHL}}{V_p}$
Iqul	$I_{q,ul}$	Algeb	$\frac{Q_{tot}}{V_p}$
Ipmax	$I_{pmax}$	Algeb	$(1 - z_l^{SOClim}) \left( I_{alim} s_1^{SW_{PQ}} + \sqrt{I_{pmax0}^2 s_0^{SW_{PQ}}} \right)$
Iqmax	$I_{qmax}$	Algeb	$I_{alim} s_0^{SW_{PQ}} + \sqrt{I_{qmax0}^2 s_1^{SW_{PQ}}}$
Ipcmd_x	$x_{Ipcmd}$	Algeb	$I_{p,ul}$
Ipcmd_y	$y_{Ipcmd}$	Algeb	$I_{pmax} z_u^{lim_{Ipcmd}} (F_{fh} F_{fl} F_{vh} F_{vl} z_{rec} - z_{rec} + 1) + x_{Ipcmd} z_i^{lim_{Ipcmd}} (F_{fh} F_{fl} F_{vh} F_{vl} z_{rec})$
Iqcmd_x	$x_{Iqcmd}$	Algeb	$I_{q,ul}$
Iqcmd_y	$y_{Iqcmd}$	Algeb	$-I_{qmax} z_l^{lim_{Iqcmd}} (F_{fh} F_{fl} F_{vh} F_{vl} z_{rec} - z_{rec} + 1) + I_{qmax} z_u^{lim_{Iqcmd}} (F_{fh} F_{fl} F_{vh} F_{vl} z_{rec})$
Ipmin	$I_{pmin}$	Algeb	$(z_u^{SOClim} - 1) \left( I_{alim} s_1^{SW_{PQ}} + \sqrt{I_{pmax0}^2 s_0^{SW_{PQ}}} \right)$
a	$\theta$	ExtAlgeb	
v	$V$	ExtAlgeb	
f	$f$	ExtAlgeb	

## Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
Ipout_y	$y_{Ipout}$	State	$1.0y_{Ipcmd} - y_{Ipout}$	$T_{ip}$
Iqout_y	$y_{Iqout}$	State	$1.0y_{Iqcmand} - y_{Iqout}$	$T_{iq}$
pIG_y	$y_{pIG}$	State	$\frac{S_{b,sys} \left( -H_C V y_{Ipout} z_1^{LTN} - \frac{V y_{Ipout} z_0^{LTN}}{H_D} \right)}{3600 E_n}$	$T_f$
SOC	$SOC$	AliasState	0	

## Algebraic Equations

Name	Sym-	Type	RHS of Equation "0 = g(x, y)"
	bol		
fHz	$f_{Hz}$	Al- geb	$ff_n - f_{Hz}$
Ffl	$F_{fl}$	Al- geb	$-F_{fl} + K_{ft01}z_i^{FL1} (f_{Hz} - f_{t0}) + z_u^{FL1}$
Ffh	$F_{fh}$	Al- geb	$-F_{fh} + z_i^{FL2} (K_{ft23} (-f_{Hz} + f_{t2}) + 1) + z_l^{FL2}$
Fdev	$f_{dev}$	Al- geb	$f_n - f_{Hz} - f_{dev}$
DB_y	$y_{DB}$	Al- geb	$D_{dn} \left( f_{dev} z_u^{db_{DB}} + z_l^{db_{DB}} (-f_{dbd} + f_{dev}) \right) - y_{DB}$
Fvl	$F_{vl}$	Al- geb	$-F_{vl} + K_{vt01}z_i^{VL1} (V - V_{t0}) + z_u^{VL1}$
Fvh	$F_{vh}$	Al- geb	$-F_{vh} + z_i^{VL2} (K_{vt23} (-V + V_{t2}) + 1) + z_l^{VL2}$
vp	$V_p$	Al- geb	$V z_i^{VLo} - V_p + 0.01 z_l^{VLo}$
Pext	$P_{ext}$	Al- geb	$P_{ext0}u - P_{ext}$
Pref	$P_{ref}$	Al- geb	$P_{ref0}u - P_{ref}$
Psum	$P_{tot}$	Al- geb	$-P_{tot} + u (P_{ext} + P_{ref} + y_{DB})$
Qdrp	$Q_{drp}$	Al- geb	$-Q_{drp} + dq/dvuz_i^{VQ2} (-V_{comp} + v_1) + q_{mn}z_u^{VQ2} + q_{mx}uz_l^{VQ1} + uz_i^{VQ1} (dq/dv (-V_{comp} + V_{qu}) + q_{mx})$
Qref	$Q_{ref}$	Al- geb	$Q_{ref0}u - Q_{ref}$
Qsum	$Q_{tot}$	Al- geb	$-Q_{tot} + u (Q_{drp} + Q_{ref})$
Ipul	$I_{p,ul}$	Al- geb	$-I_{p,ul} + \frac{P_{tot}z_i^{PHL} + p_{mn}z_l^{PHL} + p_{mx}z_u^{PHL}}{V_p}$
Iqul	$I_{q,ul}$	Al- geb	$-I_{q,ul} + \frac{Q_{tot}}{V_p}$
Ip- max	$I_{pmax}$	Al- geb	$-I_{pmax} + (1 - z_l^{SOClim}) \left( I_{alim}s_1^{SW_{PQ}} + \sqrt{I_{pmax}^2 s_0^{SW_{PQ}}} \right)$
Iq- max	$I_{qmax}$	Al- geb	$I_{alim}s_0^{SW_{PQ}} - I_{qmax} + \sqrt{I_{qmax}^2 s_1^{SW_{PQ}}}$
Ipcmc	$x_{Ipcm}$	Al- geb	$I_{p,ul} - x_{Ipcmd}$
Ipcmc	$y_{Ipcm}$	Al- geb	$I_{pmax} z_u^{\lim_{Ipcmd}} (F_{fh} F_{fl} F_{vh} F_{vl} z_{rec} - z_{rec} + 1) + x_{Ipcmd} z_i^{\lim_{Ipcmd}} (F_{fh} F_{fl} F_{vh} F_{vl} z_{rec} - z_{rec} + 1) - y_{Ipcmd}$
Iqcmci	$x_{Iqcm}$	Al- geb	$I_{q,ul} - x_{Iqcmd}$
Iqcmci	$y_{Iqcm}$	Al- geb	$-I_{qmax} z_l^{\lim_{Iqcmd}} (F_{fh} F_{fl} F_{vh} F_{vl} z_{rec} - z_{rec} + 1) + I_{qmax} z_u^{\lim_{Iqcmd}} (F_{fh} F_{fl} F_{vh} F_{vl} z_{rec} - z_{rec} + 1) + x_{Iqcmd} z_i^{\lim_{Iqcmd}} (F_{fh} F_{fl} F_{vh} F_{vl} z_{rec} - z_{rec} + 1) - y_{Iqcmd}$

## Services

Name	Symbol	Equation	Type
pref0	$P_{ref0}$	$P_{0s}\gamma_p$	ConstService
qref0	$Q_{ref0}$	$Q_{0s}\gamma_q$	ConstService
Kft01	$K_{ft01}$	$\frac{1}{-f_{t0}+f_{t1}}$	ConstService
Kft23	$K_{ft23}$	$\frac{1}{-f_{t2}+f_{t3}}$	ConstService
Kvt01	$K_{vt01}$	$\frac{1}{-V_{t0}+V_{t1}}$	ConstService
Kvt23	$K_{vt23}$	$\frac{1}{-V_{t2}+V_{t3}}$	ConstService
Pext0	$P_{ext0}$	0	ConstService
Vcomp	$V_{comp}$	$ Ve^{i\theta} + ix_c(y_{Ipout} + iy_{Iqout}) $	VarService
Vqu	$V_{qu}$	$v_1 - \frac{Q_{ref0}-q_{mn}}{dq/dv}$	ConstService
Vql	$V_{ql}$	$v_0 + \frac{-Q_{ref0}+q_{mx}}{dq/dv}$	ConstService
Ipmаксsq	$I_{pmax}^2$	FixPiecewise $\left( \left( 0, I_{alim}^2 - (y_{Ipcmd})^2 \leq 0 \right), \left( I_{alim}^2 - (y_{Ipcmd})^2, \text{True} \right) \right)$	VarService
Ip- maxsq0	$I_{pmax0}^2$	FixPiecewise $\left( \left( 0, I_{alim}^2 - \frac{Q_{ref0}^2 u^2}{V^2} \leq 0 \right), \left( I_{alim}^2 - \frac{Q_{ref0}^2 u^2}{V^2}, \text{True} \right) \right)$	ConstService
Iq максsq	$I_{qmax}^2$	FixPiecewise $\left( \left( 0, I_{alim}^2 - (y_{Ipcmd})^2 \leq 0 \right), \left( I_{alim}^2 - (y_{Ipcmd})^2, \text{True} \right) \right)$	VarService
Iq- maxsq0	$I_{qmax0}^2$	FixPiecewise $\left( \left( 0, I_{alim}^2 - \frac{P_{ref0}^2 u^2}{V^2} \leq 0 \right), \left( I_{alim}^2 - \frac{P_{ref0}^2 u^2}{V^2}, \text{True} \right) \right)$	ConstService

## 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

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)
ad-just_upper		1	adjust upper limit	(0, 1)
plim	$P_{lim}$	1	enable input power limit check bound by [0, pmx]	(0, 1)

## 5.8.4 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  $pmn \leq pmx * pcap$ .

### 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
dqv	$dq/dv$	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,pow
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
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

continues on next page

Table 9 – continued from previous page

Name	Symbol	Description	Default	Unit	Properties
gammap	$\gamma_p$	Ratio of PVD1.pref0 w.r.t to that of static PV	1		
gammaq	$\gamma_q$	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		

## Variables

Name	Symbol	Type	Description	Unit	Properties
Ipout_y	$y_{Ipout}$	State	State in lag transfer function	v_str	
Iqout_y	$y_{Iqout}$	State	State in lag transfer function	v_str	
pIG_y	$y_{pIG}$	State	Integrator output	v_str	
SOC	$SOC$	AliasState	Alias for state of charge		
fHz	$f_{Hz}$	Algeb	frequency in Hz	Hz	v_str
Ffl	$F_{fl}$	Algeb	Coeff. for under frequency	v_str	
Ffh	$F_{fh}$	Algeb	Coeff. for over frequency	v_str	
Fdev	$f_{dev}$	Algeb	Frequency deviation	Hz	v_str
DB_y	$y_{DB}$	Algeb	Deadband type 1 output	v_str	
Fvl	$F_{vl}$	Algeb	Coeff. for under voltage	v_str	
Fvh	$F_{vh}$	Algeb	Coeff. for over voltage	v_str	
vp	$V_p$	Algeb	Sensed positive voltage	v_str	
Pext	$P_{ext}$	Algeb	External power signal (for AGC)	v_str	
Pref	$P_{ref}$	Algeb	Reference power signal (for scheduling setpoint)	v_str	
Psum	$P_{tot}$	Algeb	Sum of P signals	v_str	
Qdrp	$Q_{drp}$	Algeb	External power signal (for AGC)	v_str	
Qref	$Q_{ref}$	Algeb	Reference power signal (for scheduling setpoint)	v_str	
Qsum	$Q_{tot}$	Algeb	Sum of Q signals	v_str	
Ipul	$I_{p,ul}$	Algeb	Ipcmd before Ip hard limit	v_str	
Iqul	$I_{q,ul}$	Algeb	Iqcmd before Iq hard limit	v_str	
Ipmax	$I_{pmax}$	Algeb	Upper limit of Ip	v_str	
Iqmax	$I_{qmax}$	Algeb	Upper limit of Iq	v_str	
Ipcmd_x	$x_{Ipcmd}$	Algeb	Value before limiter	v_str	
Ipcmd_y	$y_{Ipcmd}$	Algeb	Output after limiter and post gain	v_str	
Iqcmd_x	$x_{Iqcmd}$	Algeb	Value before limiter	v_str	

continues on next page

Table 10 – continued from previous page

Name	Symbol	Type	Description	Unit	Properties
Iqcmd_y	$y_{Iqcmd}$	Algeb	Output after limiter and post gain	v_str	
Ipmin	$Ipmin$	Algeb	Minimum value of Ip	v_str	
PHLup	$PHL_{upper}$	Algeb	PHL upper limit	v_str	
a	$\theta$	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

Name	Symbol	Type	Initial Value
Ipout_y	$y_{Ipout}$	State	$1.0y_{Ipcmd}$
Iqout_y	$y_{Iqout}$	State	$1.0y_{Iqcmd}$
pIG_y	$y_{pIG}$	State	$SOC_{init}$
SOC	$SOC$	AliasState	
fHz	$f_{Hz}$	Algeb	$f_n$
Ffl	$F_{fl}$	Algeb	$K_{ft01}z_i^{FL1}(f_{Hz} - f_{t0}) + z_u^{FL1}$
Ffh	$F_{fh}$	Algeb	$z_i^{FL2}(K_{ft23}(-f_{Hz} + f_{t2}) + 1) + z_l^{FL2}$
Fdev	$f_{dev}$	Algeb	$f_n - f_{Hz}$
DB_y	$y_{DB}$	Algeb	$D_{dn} \left( f_{dev}z_u^{db_{DB}} + z_l^{db_{DB}} (-f_{dbd} + f_{dev}) \right)$
Fvl	$F_{vl}$	Algeb	$K_{vt01}z_i^{VL1}(V - V_{t0}) + z_u^{VL1}$
Fvh	$F_{vh}$	Algeb	$z_i^{VL2}(K_{vt23}(-V + V_{t2}) + 1) + z_l^{VL2}$
vp	$V_p$	Algeb	$Vz_i^{VL0} + 0.01z_l^{VL0}$
Pext	$P_{ext}$	Algeb	$P_{ext0}u$
Pref	$P_{ref}$	Algeb	$P_{ref0}u$
Psum	$P_{tot}$	Algeb	$u(P_{ext} + P_{ref} + y_{DB})$
Qdrp	$Q_{drp}$	Algeb	$dq/dvuz_i^{VQ2}(-V_{comp} + v_1) + q_{mn}z_u^{VQ2} + q_{mx}uz_l^{VQ1} + uz_i^{VQ1}(dq/dv(-V_{comp} + v_1))$
Qref	$Q_{ref}$	Algeb	$Q_{ref0}u$
Qsum	$Q_{tot}$	Algeb	$u(Q_{ref0} + dq/dvuz_i^{VQ2}(-V_{comp} + v_1) + q_{mn}z_u^{VQ2} + q_{mx}uz_l^{VQ1} + uz_i^{VQ1}(dq/dv(-V_{comp} + v_1)))$
Ipul	$I_{p,ul}$	Algeb	$\frac{PHL_{upper}z_u^{PHL2} + P_{tot}z_i^{PHL2} + p_{mn}z_l^{PHL2}}{V_p}$
Iqul	$I_{q,ul}$	Algeb	$\frac{Q_{tot}}{V_p}$
Ipmax	$I_{pmax}$	Algeb	$(1 - z_l^{SOClim}) \left( I_{alim}s_1^{SW_{PQ}} + \sqrt{I_{pmax0}^2 s_0^{SW_{PQ}}} \right)$
Iqmax	$I_{qmax}$	Algeb	$I_{alim}s_0^{SW_{PQ}} + \sqrt{I_{qmax0}^2 s_1^{SW_{PQ}}}$
Ipcmd_x	$x_{Ipcmd}$	Algeb	$I_{p,ul}$
Ipcmd_y	$y_{Ipcmd}$	Algeb	$I_{pmax}z_u^{lim_{Ipcmd}}(F_{fh}F_{fl}F_{vh}F_{vl}z_{rec} - z_{rec} + 1) + x_{Ipcmd}z_i^{lim_{Ipcmd}}(F_{fh}F_{fl}F_{vh}F_{vl}z_{rec} - z_{rec} + 1)$
Iqcmd_x	$x_{Iqcmd}$	Algeb	$I_{q,ul}$
Iqcmd_y	$y_{Iqcmd}$	Algeb	$-I_{qmax}z_l^{lim_{Iqcmd}}(F_{fh}F_{fl}F_{vh}F_{vl}z_{rec} - z_{rec} + 1) + I_{qmax}z_u^{lim_{Iqcmd}}(F_{fh}F_{fl}F_{vh}F_{vl}z_{rec} - z_{rec} + 1)$
Ipmin	$Ipmin$	Algeb	$(z_u^{SOClim} - 1) \left( I_{alim}s_1^{SW_{PQ}} + \sqrt{I_{pmax0}^2 s_0^{SW_{PQ}}} \right)$
PHLup	$PHL_{upper}$	Algeb	$p_{cap}p_{mx}$

Table 11 – continued from previous page

Name	Symbol	Type	Initial Value
a	$\theta$	ExtAlgeb	
v	$V$	ExtAlgeb	
f	$f$	ExtAlgeb	

## Differential Equations

Name	Symbol	Type	RHS of Equation "T' x' = f(x, y)"	T (LHS)
Ipout_y	$y_{Ipout}$	State	$1.0y_{Ipout} - y_{Ipout}$	$T_{ip}$
Iqout_y	$y_{Iqout}$	State	$1.0y_{Iqout} - y_{Iqout}$	$T_{iq}$
pIG_y	$y_{pIG}$	State	$\frac{S_{b,sys} \left( -H_C V y_{Ipout} z_1^{LTN} - \frac{V y_{Ipout} z_0^{LTN}}{H_D} \right)}{3600 E_n}$	$T_f$
SOC	$SOC$	AliasState	0	

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
fHz	$f_{Hz}$	Algеб	$ff_n - f_{Hz}$
Ffl	$F_{fl}$	Algеб	$-F_{fl} + K_{ft01}z_i^{FL1} (f_{Hz} - f_{t0}) + z_u^{FL1}$
Ffh	$F_{fh}$	Algеб	$-F_{fh} + z_i^{FL2} (K_{ft23} (-f_{Hz} + f_{t2}) + 1) + z_l^{FL2}$
Fdev	$f_{dev}$	Algеб	$f_n - f_{Hz} - f_{dev}$
DB_y	$y_{DB}$	Algеб	$D_{dn} \left( f_{dev}z_u^{db_{DB}} + z_l^{db_{DB}} (-f_{dbd} + f_{dev}) \right) - y_{DB}$
Fvl	$F_{vl}$	Algеб	$-F_{vl} + K_{vt01}z_i^{VL1} (V - V_{t0}) + z_u^{VL1}$
Fvh	$F_{vh}$	Algеб	$-F_{vh} + z_i^{VL2} (K_{vt23} (-V + V_{t2}) + 1) + z_l^{VL2}$
vp	$V_p$	Algеб	$V z_i^{VL0} - V_p + 0.01 z_l^{VL0}$
Pext	$P_{ext}$	Algеб	$P_{ext0}u - P_{ext}$
Pref	$P_{ref}$	Algеб	$P_{ref0}u - P_{ref}$
Psum	$P_{tot}$	Algеб	$-P_{tot} + u (P_{ext} + P_{ref} + y_{DB})$
Qdrp	$Q_{drp}$	Algеб	$-Q_{drp} + dq/dv u z_i^{VQ2} (-V_{comp} + v_1) + q_{mn} z_u^{VQ2} + q_{mx} u z_l^{VQ1} + u z_i^{VQ1} (dq/dv (-V_{comp} + V_{qu}) + q_{mx})$
Qref	$Q_{ref}$	Algеб	$Q_{ref0}u - Q_{ref}$
Qsum	$Q_{tot}$	Algеб	$-Q_{tot} + u (Q_{drp} + Q_{ref})$
Ipul	$I_{p,ul}$	Algеб	$-I_{p,ul} + \frac{PHL_{upper} z_u^{PHL2} + P_{tot} z_i^{PHL2} + p_{mn} z_l^{PHL2}}{V_p}$
Iqul	$I_{q,ul}$	Algеб	$-I_{q,ul} + \frac{Q_{tot}}{V_p}$
Ipmax	$I_{pmax}$	Algеб	$-I_{pmax} + (1 - z_l^{SOClim}) \left( I_{alim} s_1^{SW_{PQ}} + \sqrt{I_{pmax}^2} s_0^{SW_{PQ}} \right)$
Iqmax	$I_{qmax}$	Algеб	$I_{alim} s_0^{SW_{PQ}} - I_{qmax} + \sqrt{I_{qmax}^2} s_1^{SW_{PQ}}$
Ipcmc	$x_{Ipcmd}$	Algеб	$I_{p,ul} - x_{Ipcmd}$
Ipcmc	$y_{Ipcmd}$	Algеб	$I_{pmax} z_u^{lim_{Ipcmd}} (F_{fh} F_{fl} F_{vh} F_{vl} z_{rec} - z_{rec} + 1) + x_{Ipcmd} z_i^{lim_{Ipcmd}} (F_{fh} F_{fl} F_{vh} F_{vl} z_{rec} - z_{rec} + 1) - y_{Ipcmd}$
Iqcmc	$x_{Iqcmd}$	Algеб	$I_{q,ul} - x_{Iqcmd}$
Iqcmc	$y_{Iqcmd}$	Algеб	$-I_{qmax} z_l^{lim_{Iqcmd}} (F_{fh} F_{fl} F_{vh} F_{vl} z_{rec} - z_{rec} + 1) + I_{qmax} z_u^{lim_{Iqcmd}} (F_{fh} F_{fl} F_{vh} F_{vl} z_{rec} - z_{rec} + 1) + x_{Iqcmd} z_i^{lim_{Iqcmd}} (F_{fh} F_{fl} F_{vh} F_{vl} z_{rec} - z_{rec} + 1) - y_{Iqcmd}$

## Services

Name	Symbol	Equation	Type
pref0	$P_{ref0}$	$P_{0s}\gamma_p$	ConstService
qref0	$Q_{ref0}$	$Q_{0s}\gamma_q$	ConstService
Kft01	$K_{ft01}$	$\frac{1}{-f_{t0}+f_{t1}}$	ConstService
Kft23	$K_{ft23}$	$\frac{1}{-f_{t2}+f_{t3}}$	ConstService
Kvt01	$K_{vt01}$	$\frac{1}{-V_{t0}+V_{t1}}$	ConstService
Kvt23	$K_{vt23}$	$\frac{1}{-V_{t2}+V_{t3}}$	ConstService
Pext0	$P_{ext0}$	0	ConstService
Vcomp	$V_{comp}$	$ Ve^{i\theta} + ix_c(y_{Ipout} + iy_{Iqout}) $	VarService
Vqu	$V_{qu}$	$v_1 - \frac{Q_{ref0}-q_{mn}}{dq/dv}$	ConstService
Vql	$V_{ql}$	$v_0 + \frac{-Q_{ref0}+q_{mx}}{dq/dv}$	ConstService
Ipmаксsq	$I_{pmax}^2$	FixPiecewise $\left( \left( 0, I_{alim}^2 - (y_{Ipcmd})^2 \leq 0 \right), \left( I_{alim}^2 - (y_{Ipcmd})^2, \text{True} \right) \right)$	VarService
Ip- maxsq0	$I_{pmax0}^2$	FixPiecewise $\left( \left( 0, I_{alim}^2 - \frac{Q_{ref0}^2 u^2}{V^2} \leq 0 \right), \left( I_{alim}^2 - \frac{Q_{ref0}^2 u^2}{V^2}, \text{True} \right) \right)$	ConstService
Iq максsq	$I_{qmax}^2$	FixPiecewise $\left( \left( 0, I_{alim}^2 - (y_{Ipcmd})^2 \leq 0 \right), \left( I_{alim}^2 - (y_{Ipcmd})^2, \text{True} \right) \right)$	VarService
Iq- maxsq0	$I_{qmax0}^2$	FixPiecewise $\left( \left( 0, I_{alim}^2 - \frac{P_{ref0}^2 u^2}{V^2} \leq 0 \right), \left( I_{alim}^2 - \frac{P_{ref0}^2 u^2}{V^2}, \text{True} \right) \right)$	ConstService

## 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	$Ipcmd$	GainLimiter	Ip with limiter and coeff.
Iqcmd	$Iqcmd$	GainLimiter	Iq with limiter and coeff.
Ipout	$Ipout$	Lag	Output Ip filter
Iqout	$Iqout$	Lag	Output Iq filter
pIG	$pIG$	Integrator	State of charge

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)
ad-		1	adjust upper limit	(0, 1)
just_upper				
plim	$P_{lim}$	1	enable input power limit check bound by [0, pmx]	(0, 1)

## 5.9 DGProtection

Protection model for DG.

Common Parameters: u, name

Available models: *DGPRCT1*, *DGPRCTExt*

### 5.9.1 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)  $P_{sum}$  and  $Q_{sum}$  will decrease to zero immediately when frequency/voltage protection flag is raised. Once the lock is released,  $P_{sum}$  and  $Q_{sum}$  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  $recflag = 0$ .

$f_{en}$  and  $V_{en}$  are protection enabling parameters. 1/0 is on/off.

$ue$  is lock flag signal.

It should be noted that, the lock only lock the  $f_{Hz}$  (frequency read value) of DG model. The source values (which come from BusFreqf remain unchanged.)

Protection sensors (e.g., IAWfl1) are instances of `IntegratorAntiWindup`. All the protection sensors will be reset after  $ue$  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  $f_{l3} < f_{l2} < f_{l1} < f_{u1} < f_{u2} < f_{u3}$ , and  $u_{l4} < u_{l3} < u_{l2} < u_{l1} < u_{u1} < u_{u2} < u_{u3}$ .

Default settings:

Frequency (Hz):

$(f_{l3}, f_{l2})$ ,  $Tf_{l2} [(50.0, 57.5), 10s]$

$(f_{l2}, f_{l1})$ ,  $Tf_{l1} [(57.5, 59.2), 300s]$

$(f_{u1}, f_{u2})$ ,  $Tf_{u1} [(60.5, 61.5), 300s]$

$(f_{u2}, f_{u3})$ ,  $Tf_{u2} [(61.5, 70.0), 10s]$

Voltage (p.u.):

$(v_{l4}, v_{l3})$ ,  $Tv_{l3} [(0.10, 0.45), 0.16s]$

$(v_{l3}, v_{l2})$ ,  $Tv_{l2} [(0.45, 0.60), 1s]$

$(v_{l2}, v_{l1})$ ,  $Tv_{l1} [(0.60, 0.88), 2s]$

$(v_{u1}, v_{u2})$ ,  $Tv_{u1} [(1.10, 1.20), 1s]$

$(v_{u2}, v_{u3})$ ,  $Tv_{u2} [(1.20, 2.00), 0.16s]$

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](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	<i>u</i>	connection status	1		<i>bool</i>
name		device name			
dev		idx of the target device			mandatory
busfreq		Target device interface bus measurement device idx			
fen	<i>fen</i>	Frequency deviation protection enable. 1 for enable, 0 for disable.	1		
Ven	<i>Ven</i>	Voltage deviation protection enable. 1 for enable, 0 for disable.	0		
fl3	<i>fl3</i>	Under frequency shadding point 3	50	<i>Hz</i>	
fl2	<i>fl2</i>	Over frequency shadding point 2	57.500	<i>Hz</i>	
fl1	<i>fl1</i>	Under frequency shadding point 1	59.200	<i>Hz</i>	
fu1	<i>fu1</i>	Over frequency shadding point 1	60.500	<i>Hz</i>	
fu2	<i>fu2</i>	Over frequency shadding point 2	61.500	<i>Hz</i>	
fu3	<i>fu3</i>	Over frequency shadding point 3	70	<i>Hz</i>	
Tfl1	<i>Tfl1</i>	Stand time for (fl2, fl1)	300		non_negative
Tfl2	<i>Tfl2</i>	Stand time for (fl3, fl2)	10		non_negative
Tfu1	<i>Tfu1</i>	Stand time for (fu1, fu2)	300		non_negative
Tfu2	<i>Tfu2</i>	Stand time for (fu2, fu3)	10		non_negative
vl4	<i>vl4</i>	Under voltage shadding point 4	0.100	<i>p.u.</i>	
vl3	<i>vl3</i>	Under voltage shadding point 3	0.450	<i>p.u.</i>	
vl2	<i>vl2</i>	Under voltage shadding point 2	0.600	<i>p.u.</i>	
vl1	<i>vl1</i>	Under voltage shadding point 1	0.880	<i>p.u.</i>	
vu1	<i>vu1</i>	Over voltage shadding point 1	1.100	<i>p.u.</i>	
vu2	<i>vu2</i>	Over voltage shadding point 2	1.200	<i>p.u.</i>	
vu3	<i>vu3</i>	Over voltage shadding point 3	2	<i>p.u.</i>	
Tvl1	<i>Tvl1</i>	Stand time for (vl2, vl1)	2		non_negative
Tvl2	<i>Tvl2</i>	Stand time for (vl3, vl2)	1		non_negative
Tvl3	<i>Tvl3</i>	Stand time for (vl4, vl3)	0.160		non_negative
Tvu1	<i>Tvu1</i>	Stand time for (vu1, vu2)	1		non_negative
Tvu2	<i>Tvu2</i>	Stand time for (vu2, vu3)	0.160		non_negative
Tres		Integrator reset time	0.050		
bus			0		
fn			0		

## Variables

Name	Symbol	Type	Description	Unit	Properties
IAWfl1_y	$y_{IAWfl1}$	State	AW Integrator output	v_str	
IAWfl2_y	$y_{IAWfl2}$	State	AW Integrator output	v_str	
IAWfu1_y	$y_{IAWfu1}$	State	AW Integrator output	v_str	
IAWfu2_y	$y_{IAWfu2}$	State	AW Integrator output	v_str	
IAWVl1_y	$y_{IAWVl1}$	State	AW Integrator output	v_str	
IAWVl2_y	$y_{IAWVl2}$	State	AW Integrator output	v_str	
IAWVl3_y	$y_{IAWVl3}$	State	AW Integrator output	v_str	
IAVVu1_y	$y_{IAVVu1}$	State	AW Integrator output	v_str	
IAVVu2_y	$y_{IAVVu2}$	State	AW Integrator output	v_str	
fHz	$f_{Hz}$	Algeb	frequency in Hz	v_str	
dsum	$d_{tot}$	Algeb	lock signal summation	v_str	
ue	$ue$	Algeb	lock flag	v_str	
f	$f$	ExtAlgeb	DG frequency read value	p.u.	
fin	$fin$	ExtAlgeb	original f from DG		
fHzl	$f_{Hzl}$	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	$y_{IAWfl1}$	State	0
IAWfl2_y	$y_{IAWfl2}$	State	0
IAWfu1_y	$y_{IAWfu1}$	State	0
IAWfu2_y	$y_{IAWfu2}$	State	0
IAWVl1_y	$y_{IAWVl1}$	State	0
IAWVl2_y	$y_{IAWVl2}$	State	0
IAWVl3_y	$y_{IAWVl3}$	State	0
IAWVu1_y	$y_{IAWVu1}$	State	0
IAWVu2_y	$y_{IAWVu2}$	State	0
fHz	$f_{Hz}$	Algeb	$ffn$
dsum	$d_{tot}$	Algeb	0
ue	$ue$	Algeb	0
f	$f$	ExtAlgeb	
fin	$fin$	ExtAlgeb	
fHzl	$f_{Hzl}$	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	$y_{IAWfl1}$	State	$-\frac{T_{fl1}res}{Tres} + z_i^{Lfl1} \cdot (1 - res)$	1
IAWfl2_y	$y_{IAWfl2}$	State	$-\frac{T_{fl2}res}{Tres} + z_i^{Lfl2} \cdot (1 - res)$	1
IAWfu1_y	$y_{IAWfu1}$	State	$-\frac{T_{fu1}res}{Tres} + z_i^{Lfu1} \cdot (1 - res)$	1
IAWfu2_y	$y_{IAWfu2}$	State	$-\frac{T_{fu2}res}{Tres} + z_i^{Lfl2} \cdot (1 - res)$	1
IAWVl1_y	$y_{IAWVl1}$	State	$-\frac{T_{vl1}res}{Tres} + z_i^{LVL1} \cdot (1 - res)$	1
IAWVl2_y	$y_{IAWVl2}$	State	$-\frac{T_{vl2}res}{Tres} + z_i^{LVL2} \cdot (1 - res)$	1
IAWVl3_y	$y_{IAWVl3}$	State	$-\frac{T_{vl3}res}{Tres} + z_i^{LVL2} \cdot (1 - res)$	1
IAWVu1_y	$y_{IAWVu1}$	State	$-\frac{T_{vu1}res}{Tres} + z_i^{LVU1} \cdot (1 - res)$	1
IAWVu2_y	$y_{IAWVu2}$	State	$-\frac{T_{vu2}res}{Tres} + z_i^{LVU2} \cdot (1 - res)$	1

## Algebraic Equations

Nam bol	Sym	Type	RHS of Equation "0 = g(x, y)"
fHz	$f_{Hz}$	Al- geb	$ffn - f_{Hz}$
dsum	$d_{tot}$	Al- geb	$Ven(z_i^{LVi1}z_u^{lim_{IAWVi1}} + z_i^{LVi2}z_u^{lim_{IAWVi2}} + z_i^{LVi3}z_u^{lim_{IAWVi3}} + z_i^{LVu1}z_u^{lim_{IAWVu1}} + z_i^{LVu2}z_u^{lim_{IAWVu2}}) + d_{tot} + fen(z_i^{Lfl1}z_u^{lim_{IAWfl1}} + z_i^{Lfl2}z_u^{lim_{IAWfl2}} + z_i^{Lfu1}z_u^{lim_{IAWfu1}} + z_i^{Lfu2}z_u^{lim_{IAWfu2}})$
ue	$ue$	Al- geb	$-ue + z_u^{Ldsum}$
f	$f$	Ex- tAl- geb	0
fin	$fin$	Ex- tAl- geb	0
fHzl	$f_{Hzl}$	Ex- tAl- geb	$-ffnue$
Pext	$P_{ext}$	Ex- tAl- geb	0
Pref	$P_{ref}$	Ex- tAl- geb	0
Pdrp	$P_{drp}$	Ex- tAl- geb	0
Psun	$P_{sun}$	Ex- tAl- geb	$-ue(P_{drp} + P_{ext} + P_{ref})$
Qdrp	$Q_{drp}$	Ex- tAl- geb	0
Qref	$Q_{ref}$	Ex- tAl- geb	0
Qsun	$Q_{sun}$	Ex- tAl- geb	$-ue(Q_{drp} + Q_{ref})$
v	$v$	Ex- tAl- geb	0

## Services

Name	Symbol	Equation	Type
ltu	<i>ltu</i>	0.8	ConstService
ltl	<i>ltl</i>	0.2	ConstService
zero	<i>zero</i>	0	ConstService
res	<i>res</i>	0	ExtendedEvent

## Discretes

Name	Symbol	Type	Info
Ldsum	<i>Ldsum</i>	Limiter	lock signal comparer, zu is to act
Lfl1	<i>Lfl1</i>	Limiter	Frequency comparer for (fl3, fl1)
Lfl2	<i>Lfl2</i>	Limiter	Frequency comparer for (fl3, fl2)
Lfu1	<i>Lfu1</i>	Limiter	Frequency comparer for (fu1, fu3)
Lfu2	<i>Lfu2</i>	Limiter	Frequency comparer for (fu2, fu3)
IAWfl1_lim	<i>lim<sub>IAWfl1</sub></i>	AntiWindup	Limiter in integrator
IAWfl2_lim	<i>lim<sub>IAWfl2</sub></i>	AntiWindup	Limiter in integrator
IAWfu1_lim	<i>lim<sub>IAWfu1</sub></i>	AntiWindup	Limiter in integrator
IAWfu2_lim	<i>lim<sub>IAWfu2</sub></i>	AntiWindup	Limiter in integrator
Lvl1	<i>Lvl1</i>	Limiter	Voltage comparer for (vl4, vl1)
Lvl2	<i>Lvl2</i>	Limiter	Voltage comparer for (vl4, vl2)
Lvl3	<i>Lvl3</i>	Limiter	Voltage comparer for (vl4, vl3)
LVu1	<i>LVu1</i>	Limiter	Voltage comparer for (vu1, vu3)
LVu2	<i>LVu2</i>	Limiter	Voltage comparer for (vu2, vu3)
IAWVl1_lim	<i>lim<sub>IAWVl1</sub></i>	AntiWindup	Limiter in integrator
IAWVl2_lim	<i>lim<sub>IAWVl2</sub></i>	AntiWindup	Limiter in integrator
IAWVl3_lim	<i>lim<sub>IAWVl3</sub></i>	AntiWindup	Limiter in integrator
IAWVu1_lim	<i>lim<sub>IAWVu1</sub></i>	AntiWindup	Limiter in integrator
IAWVu2_lim	<i>lim<sub>IAWVu2</sub></i>	AntiWindup	Limiter in integrator

## Blocks

Name	Symbol	Type	Info
IAWfl1	<i>IAWfl1</i>	IntegratorAntiWindup	condition check for (fl3, fl1)
IAWfl2	<i>IAWfl2</i>	IntegratorAntiWindup	condition check for (fl3, fl2)
IAWfu1	<i>IAWfu1</i>	IntegratorAntiWindup	condition check for (fu1, fu3)
IAWfu2	<i>IAWfu2</i>	IntegratorAntiWindup	condition check for (fu2, fu3)
IAWVl1	<i>IAWVl1</i>	IntegratorAntiWindup	condition check for (Vi3, Vi1)
IAWVl2	<i>IAWVl2</i>	IntegratorAntiWindup	condition check for (Vi3, Vi2)
IAWVl3	<i>IAWVl3</i>	IntegratorAntiWindup	condition check for (Vi3, Vi2)
IAVVu1	<i>IAVVu1</i>	IntegratorAntiWindup	condition check for (Vu1, Vu3)
IAVVu2	<i>IAVVu2</i>	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)

### 5.9.2 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)  $P_{sum}$  and  $Q_{sum}$  will decrease to zero immediately when frequency/voltage protection flag is raised. Once the lock is released,  $P_{sum}$  and  $Q_{sum}$  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  $recflag = 0$ .

$f_{en}$  and  $V_{en}$  are protection enabling parameters. 1/0 is on/off.

$ue$  is lock flag signal.

It should be noted that, the lock only lock the  $f_{Hz}$  (frequency read value) of DG model. The source values (which come from  $BusFreqf$ ) remain unchanged.)

Protection sensors (e.g., IAWfl1) are instances of `IntegratorAntiWindup`. All the protection sensors will be reset after  $ue$  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  $f_{l3} < f_{l2} < f_{l1} < f_{u1} < f_{u2} < f_{u3}$ , and  $u_{l4} < u_{l3} < u_{l2} < u_{l1} < u_{u2} < u_{u3}$ .

Default settings:

Frequency (Hz):

$(f13, f12), T_{f12} [(50.0, 57.5), 10s]$

$(f12, f11), T_{f11} [(57.5, 59.2), 300s]$

$(fu1, fu2), T_{fu1} [(60.5, 61.5), 300s]$

$(fu2, fu3), T_{fu2} [(61.5, 70.0), 10s]$

Voltage (p.u.):

$(vl4, vl3), T_{vl3} [(0.10, 0.45), 0.16s]$

$(vl3, vl2), T_{vl2} [(0.45, 0.60), 1s]$

$(vl2, vl1), T_{vl1} [(0.60, 0.88), 2s]$

$(vu1, vu2), T_{vu1} [(1.10, 1.20), 1s]$

$(vu2, vu3), T_{vu2} [(1.20, 2.00), 0.16s]$

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](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		<i>bool</i>
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		
f13	$f13$	Under frequency shadding point 3	50	<i>Hz</i>	
f12	$f12$	Over frequency shadding point 2	57.500	<i>Hz</i>	
f11	$f11$	Under frequency shadding point 1	59.200	<i>Hz</i>	
fu1	$fu1$	Over frequency shadding point 1	60.500	<i>Hz</i>	
fu2	$fu2$	Over frequency shadding point 2	61.500	<i>Hz</i>	
fu3	$fu3$	Over frequency shadding point 3	70	<i>Hz</i>	
Tf11	$T_{f11}$	Stand time for (f12, f11)	300		<i>non_negative</i>
Tf12	$T_{f12}$	Stand time for (f13, f12)	10		<i>non_negative</i>
Tfu1	$T_{fu1}$	Stand time for (fu1, fu2)	300		<i>non_negative</i>

continues on next page

Table 13 – continued from previous page

Name	Symbol	Description	Default	Unit	Properties
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		
bus			0		
fn			0		

## Variables

Name	Symbol	Type	Description	Unit	Properties
IAWfl1_y	$y_{IAWfl1}$	State	AW Integrator output	v_str	
IAWfl2_y	$y_{IAWfl2}$	State	AW Integrator output	v_str	
IAWfu1_y	$y_{IAWfu1}$	State	AW Integrator output	v_str	
IAWfu2_y	$y_{IAWfu2}$	State	AW Integrator output	v_str	
IAWVl1_y	$y_{IAWVl1}$	State	AW Integrator output	v_str	
IAWVl2_y	$y_{IAWVl2}$	State	AW Integrator output	v_str	
IAWVl3_y	$y_{IAWVl3}$	State	AW Integrator output	v_str	
IAWVu1_y	$y_{IAWVu1}$	State	AW Integrator output	v_str	
IAWVu2_y	$y_{IAWVu2}$	State	AW Integrator output	v_str	
fHz	$f_{Hz}$	Algeb	frequency in Hz	v_str	
dsum	$d_{tot}$	Algeb	lock signal summation	v_str	
ue	$ue$	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	$y_{IAWfl1}$	State	0
IAWfl2_y	$y_{IAWfl2}$	State	0
IAWfu1_y	$y_{IAWfu1}$	State	0
IAWfu2_y	$y_{IAWfu2}$	State	0
IAWVl1_y	$y_{IAWVl1}$	State	0
IAWVl2_y	$y_{IAWVl2}$	State	0
IAWVl3_y	$y_{IAWVl3}$	State	0
IAWVu1_y	$y_{IAWVu1}$	State	0
IAWVu2_y	$y_{IAWVu2}$	State	0
fHz	$f_{Hz}$	Algeb	$ffn$
dsum	$d_{tot}$	Algeb	0
ue	$ue$	Algeb	0
f	$f$	ExtAlgeb	
fin	$fin$	ExtAlgeb	
fHzl	$f_{Hzl}$	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' = f(x, y)"	T (LHS)
IAWfl1_y	$y_{IAWfl1}$	State	$-\frac{T_{fl1}res}{Tres} + z_i^{Lfl1} \cdot (1 - res)$	1
IAWfl2_y	$y_{IAWfl2}$	State	$-\frac{T_{fl2}res}{Tres} + z_i^{Lfl2} \cdot (1 - res)$	1
IAWfu1_y	$y_{IAWfu1}$	State	$-\frac{T_{fu1}res}{Tres} + z_i^{Lfu1} \cdot (1 - res)$	1
IAWfu2_y	$y_{IAWfu2}$	State	$-\frac{T_{fu2}res}{Tres} + z_i^{Lfu2} \cdot (1 - res)$	1
IAWVl1_y	$y_{IAWVl1}$	State	$-\frac{T_{vl1}res}{Tres} + z_i^{LVL1} \cdot (1 - res)$	1
IAWVl2_y	$y_{IAWVl2}$	State	$-\frac{T_{vl2}res}{Tres} + z_i^{LVL2} \cdot (1 - res)$	1
IAWVl3_y	$y_{IAWVl3}$	State	$-\frac{T_{vl3}res}{Tres} + z_i^{LVL2} \cdot (1 - res)$	1
IAWVu1_y	$y_{IAWVu1}$	State	$-\frac{T_{vu1}res}{Tres} + z_i^{LVU1} \cdot (1 - res)$	1
IAWVu2_y	$y_{IAWVu2}$	State	$-\frac{T_{vu2}res}{Tres} + z_i^{LVU2} \cdot (1 - res)$	1

## Algebraic Equations

Nam bol	Sym	Type	RHS of Equation "0 = g(x, y)"
fHz	$f_{Hz}$	Al- geb	$ffn - f_{Hz}$
dsum	$d_{tot}$	Al- geb	$Ven(z_i^{LVi1}z_u^{lim_{IAWVi1}} + z_i^{LVi2}z_u^{lim_{IAWVi2}} + z_i^{LVi3}z_u^{lim_{IAWVi3}} + z_i^{LVu1}z_u^{lim_{IAWVu1}} + z_i^{LVu2}z_u^{lim_{IAWVu2}}) + d_{tot} + fen(z_i^{Lfl1}z_u^{lim_{IAWfl1}} + z_i^{Lfl2}z_u^{lim_{IAWfl2}} + z_i^{Lfu1}z_u^{lim_{IAWfu1}} + z_i^{Lfu2}z_u^{lim_{IAWfu2}})$
ue	$ue$	Al- geb	$-ue + z_u^{Ldsum}$
f	$f$	Ex- tAl- geb	0
fin	$fin$	Ex- tAl- geb	0
fHzl	$f_{Hzl}$	Ex- tAl- geb	$-ffnue$
Pext	$P_{ext}$	Ex- tAl- geb	0
Pref	$P_{ref}$	Ex- tAl- geb	0
Pdrp	$P_{drp}$	Ex- tAl- geb	0
Psun	$P_{sun}$	Ex- tAl- geb	$-ue(P_{drp} + P_{ext} + P_{ref})$
Qdrp	$Q_{drp}$	Ex- tAl- geb	0
Qref	$Q_{ref}$	Ex- tAl- geb	0
Qsun	$Q_{sun}$	Ex- tAl- geb	$-ue(Q_{drp} + Q_{ref})$

## Services

Name	Symbol	Equation	Type
ltu	<i>ltu</i>	0.8	ConstService
ltl	<i>ltl</i>	0.2	ConstService
zero	<i>zero</i>	0	ConstService
res	<i>res</i>	0	ExtendedEvent

## Discretes

Name	Symbol	Type	Info
Ldsum	<i>Ldsum</i>	Limiter	lock signal comparer, zu is to act
Lfl1	<i>Lfl1</i>	Limiter	Frequency comparer for (fl3, fl1)
Lfl2	<i>Lfl2</i>	Limiter	Frequency comparer for (fl3, fl2)
Lfu1	<i>Lfu1</i>	Limiter	Frequency comparer for (fu1, fu3)
Lfu2	<i>Lfu2</i>	Limiter	Frequency comparer for (fu2, fu3)
IAWfl1_lim	<i>lim<sub>IAWfl1</sub></i>	AntiWindup	Limiter in integrator
IAWfl2_lim	<i>lim<sub>IAWfl2</sub></i>	AntiWindup	Limiter in integrator
IAWfu1_lim	<i>lim<sub>IAWfu1</sub></i>	AntiWindup	Limiter in integrator
IAWfu2_lim	<i>lim<sub>IAWfu2</sub></i>	AntiWindup	Limiter in integrator
Lvl1	<i>Lvl1</i>	Limiter	Voltage comparer for (vl4, vl1)
Lvl2	<i>Lvl2</i>	Limiter	Voltage comparer for (vl4, vl2)
Lvl3	<i>Lvl3</i>	Limiter	Voltage comparer for (vl4, vl3)
LVu1	<i>LVu1</i>	Limiter	Voltage comparer for (vu1, vu3)
LVu2	<i>LVu2</i>	Limiter	Voltage comparer for (vu2, vu3)
IAWVl1_lim	<i>lim<sub>IAWVl1</sub></i>	AntiWindup	Limiter in integrator
IAWVl2_lim	<i>lim<sub>IAWVl2</sub></i>	AntiWindup	Limiter in integrator
IAWVl3_lim	<i>lim<sub>IAWVl3</sub></i>	AntiWindup	Limiter in integrator
IAWVu1_lim	<i>lim<sub>IAWVu1</sub></i>	AntiWindup	Limiter in integrator
IAWVu2_lim	<i>lim<sub>IAWVu2</sub></i>	AntiWindup	Limiter in integrator

## Blocks

Name	Symbol	Type	Info
IAWfl1	<i>IAWfl1</i>	IntegratorAntiWindup	condition check for (fl3, fl1)
IAWfl2	<i>IAWfl2</i>	IntegratorAntiWindup	condition check for (fl3, fl2)
IAWfu1	<i>IAWfu1</i>	IntegratorAntiWindup	condition check for (fu1, fu3)
IAWfu2	<i>IAWfu2</i>	IntegratorAntiWindup	condition check for (fu2, fu3)
IAWVl1	<i>IAWVl1</i>	IntegratorAntiWindup	condition check for (Vi3, Vi1)
IAWVl2	<i>IAWVl2</i>	IntegratorAntiWindup	condition check for (Vi3, Vi2)
IAWVl3	<i>IAWVl3</i>	IntegratorAntiWindup	condition check for (Vi3, Vi2)
IAVVu1	<i>IAVVu1</i>	IntegratorAntiWindup	condition check for (Vu1, Vu3)
IAVVu2	<i>IAVVu2</i>	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)

## 5.10 DataSeries

Group for TimeSeries models.

Available models: [TimeSeries](#)

### 5.10.1 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	De-fault	Unit	Properties
idx		unique device idx			
u	<i>u</i>	connection status	1	<i>bool</i>	
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

## Discretes

Name	Symbol	Type	Info
SW	<i>SW</i>	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)
silent		1	suppress output messages if is not zero	(0, 1)

## 5.11 DynLoad

Dynamic load group.

Common Parameters: u, name

Available models: [ZIP](#), [FLoad](#)

### 5.11.1 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	<i>bool</i>	
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
bus		retrieved bus idx	0		

#### Variables

Name	Symbol	Type	Description	Unit	Properties
a	$\theta$	ExtAlgeb			
v	$V$	ExtAlgeb			

## Initialization Equations

Name	Symbol	Type	Initial Value
a	$\theta$	ExtAlgeb	
v	$V$	ExtAlgeb	

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
a	$\theta$	ExtAlgeb	$P_{i0}V + P_{p0} + P_{z0}V^2$
v	$V$	ExtAlgeb	$Q_{i0}V + Q_{p0} + Q_{z0}V^2$

## Services

Name	Symbol	Equation	Type
kps	$K_{psum}$	$K_{pi} + K_{pp} + K_{pz}$	ConstService
kqs	$K_{qsum}$	$K_{qi} + K_{qp} + K_{qz}$	ConstService
rpp	$r_{pp}$	$\frac{K_{pp}u}{100}$	ConstService
rpi	$r_{pi}$	$\frac{K_{pi}u}{100}$	ConstService
rpz	$r_{pz}$	$\frac{K_{pz}u}{100}$	ConstService
rqp	$r_{qp}$	$\frac{K_{qp}u}{100}$	ConstService
rqi	$r_{qi}$	$\frac{K_{qi}u}{100}$	ConstService
rqz	$r_{qz}$	$\frac{K_{qz}u}{100}$	ConstService
pp0	$P_{p0}$	$P_0 r_{pp}$	ConstService
pi0	$P_{i0}$	$\frac{P_0 r_{pi}}{V_0}$	ConstService
pz0	$P_{z0}$	$\frac{P_0 r_{pz}}{V_0^2}$	ConstService
qp0	$Q_{p0}$	$Q_0 r_{qp}$	ConstService
qi0	$Q_{i0}$	$\frac{Q_0 r_{qi}}{V_0}$	ConstService
qz0	$Q_{z0}$	$\frac{Q_0 r_{qz}}{V_0^2}$	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)

## 5.11.2 FLoad

Voltage and frequency dependent load.

### Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	$u$	connection status	1		<i>bool</i>
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		
bus			0		

### Variables

Name	Symbol	Type	Description	Unit	Properties
f	$f$	ExtAlgeb			
a	$\theta$	ExtAlgeb			
v	$V$	ExtAlgeb			

### Initialization Equations

Name	Symbol	Type	Initial Value
f	$f$	ExtAlgeb	
a	$\theta$	ExtAlgeb	
v	$V$	ExtAlgeb	

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
f	$f$	ExtAlgeb	0
a	$\theta$	ExtAlgeb	$V^{ap} f^{bp} p v_0$
v	$V$	ExtAlgeb	$V^{aq} f^{ba} q v_0$

## Services

Name	Symbol	Equation	Type
pv0	$p v_0$	$\frac{P_0 V_0^{-ap} kpu}{100}$	ConstService
qv0	$q v_0$	$\frac{Q_0 V_0^{aq} kqu}{100}$	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)

## 5.12 Exciter

Exciter group for synchronous generators.

Common Parameters: u, name, syn

Common Variables: vout, vi

Available models: [EXDC2](#), [IEEEEX1](#), [ESDC1A](#), [ESDC2A](#), [EXST1](#), [ESST3A](#), [SEXS](#), [IEEET1](#), [EXAC1](#), [EXAC2](#), [EXAC4](#), [ESST4B](#), [AC8B](#), [IEEET3](#), [ESAC1A](#), [ESST1A](#), [ESAC5A](#)

### 5.12.1 EXDC2

EXDC2 model.

## Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	$u$	connection status	1	<i>bool</i>	
name		device name			
syn		Synchronous generator idx			mandatory
TR	$T_R$	Sensing time constant	0.010	<i>p.u.</i>	
TA	$T_A$	Lag time constant in anti-windup lag	0.040	<i>p.u.</i>	
TC	$T_C$	Lead time constant in lead-lag	1	<i>p.u.</i>	
TB	$T_B$	Lag time constant in lead-lag	1	<i>p.u.</i>	
TE	$T_E$	Exciter integrator time constant	0.800	<i>p.u.</i>	
TF1	$T_{F1}$	Feedback washout time constant	1	<i>p.u.</i>	non_zero
KF1	$K_{F1}$	Feedback washout gain	0.030	<i>p.u.</i>	
KA	$K_A$	Gain in anti-windup lag TF	40	<i>p.u.</i>	
KE	$K_E$	Gain added to saturation	1	<i>p.u.</i>	
VRMAX	$V_{RMAX}$	Maximum excitation limit	7.300	<i>p.u.</i>	
VRMIN	$V_{RMIN}$	Minimum excitation limit	-7.300	<i>p.u.</i>	
E1	$E_1$	First saturation point	0	<i>p.u.</i>	
SE1	$S_{E1}$	Value at first saturation point	0	<i>p.u.</i>	
E2	$E_2$	Second saturation point	1	<i>p.u.</i>	
SE2	$S_{E2}$	Value at second saturation point	1	<i>p.u.</i>	
ug	$u_g$	Generator online status	0	<i>bool</i>	
Sn	$S_m$	Rated power from generator	0	<i>MVA</i>	
Vn	$V_m$	Rated voltage from generator	0	<i>kV</i>	
bus	$bus$	Bus idx of the generators	0		

## Variables

Name	Symbol	Type	Description	Unit	Properties
vp	$V_p$	State	Voltage after saturation feedback, before speed term	p.u.	v_str
LS_y	$y_{LS}$	State	State in lag transfer function		v_str
LL_x	$x'_{LL}$	State	State in lead-lag		v_str
LA_y	$y_{LA}$	State	State in lag TF		v_str
W_x	$x'_W$	State	State in washout filter		v_str
omega	$\omega$	ExtState	Generator speed		
v	$E_{term}$	Algeb	Input to exciter (bus v or Eterm)		v_str
vout	$v_{out}$	Algeb	Exciter final output voltage		v_str
vref	$V_{ref}$	Algeb	Reference voltage input	p.u.	v_str
Se	$S_e( V_{out} )$	Algeb	saturation output		v_str
vi	$V_i$	Algeb	Total input voltages	p.u.	v_str
LL_y	$y_{LL}$	Algeb	Output of lead-lag		v_str
W_y	$y_W$	Algeb	Output of washout filter		v_str
vf	$v_f$	ExtAlgeb	Excitation field voltage to generator		
Xad-Ifd	$X_{ad}I_{fd}$	ExtAlgeb	Armature excitation current		
a	$\theta$	ExtAlgeb	Bus voltage phase angle		
vbus	$V$	ExtAlgeb	Bus voltage magnitude		

## Initialization Equations

Name	Symbol	Type	Initial Value
vp	$V_p$	State	$v_{f0}$
LS_y	$y_{LS}$	State	$1.0E_{term}$
LL_x	$x'_{LL}$	State	$V_i$
LA_y	$y_{LA}$	State	$K_A y_{LL}$
W_x	$x'_{W}$	State	$V_p$
omega	$\omega$	ExtState	
v	$E_{term}$	Algeb	$V$
vout	$v_{out}$	Algeb	$u_e v_{f0}$
vref	$V_{ref}$	Algeb	$E_{term} + V_{b0}$
Se	$S_e( V_{out} )$	Algeb	$S_{e0}$
vi	$V_i$	Algeb	$V_{b0}$
LL_y	$y_{LL}$	Algeb	$V_i$
W_y	$y_W$	Algeb	0
vf	$v_f$	ExtAlgeb	
XadIfd	$X_{ad} I_{fd}$	ExtAlgeb	
a	$\theta$	ExtAlgeb	
vbus	$V$	ExtAlgeb	

## Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
vp	$V_p$	State	$u_e (-K_E V_p - S_e( V_{out} ) V_p + y_{LA})$	$T_E$
LS_y	$y_{LS}$	State	$1.0E_{term} - y_{LS}$	$T_R$
LL_x	$x'_{LL}$	State	$V_i - x'_{LL}$	$T_B$
LA_y	$y_{LA}$	State	$K_A y_{LL} - y_{LA}$	$T_A$
W_x	$x'_{W}$	State	$V_p - x'_{W}$	$T_{F1}$
omega	$\omega$	ExtState	0	

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
v	$E_{term}$	Algeb	$-E_{term} + V$
vout	$v_{out}$	Algeb	$V_p \omega u_e - v_{out}$
vref	$V_{ref}$	Algeb	$V_{ref0} - V_{ref}$
Se	$S_e( V_{out} )$	Algeb	$B_{SAT}^q z_0^{SL} (-A_{SAT}^q + V_p)^2 - S_e( V_{out} ) V_p$
vi	$V_i$	Algeb	$-V_i + V_{ref} - y_{LS} - y_W$
LL_y	$y_{LL}$	Algeb	$T_B x'_{LL} - T_B y_{LL} + T_C (V_i - x'_{LL}) + (z_1^{LT_{LL}})^2 (-x'_{LL} + y_{LL})$
W_y	$y_W$	Algeb	$K_{F1} (V_p - x'_W) - T_{F1} y_W$
vf	$v_f$	ExtAlgeb	$u_e (-v_{f0} + v_{out})$
XadIfd	$X_{ad} I_{fd}$	ExtAlgeb	0
a	$\theta$	ExtAlgeb	0
vbus	$V$	ExtAlgeb	0

## Services

Name	Symbol	Equation	Type
ue	$u_e$	$uu_g$	ConstService
SAT_E1	$E_{SAT}^{1c}$	$E_1$	ConstService
SAT_E2	$E_{SAT}^{2c}$	$E_2 - 2z_{SAT}^{SE2} + 2$	ConstService
SAT_SE1	$SE_{SAT}^{1c}$	$S_{E1}$	ConstService
SAT_SE2	$SE_{SAT}^{2c}$	$S_{E2} - 2z_{SAT}^{SE2} + 2$	ConstService
SAT_a	$a_{SAT}$	$\sqrt{\frac{E_{SAT}^{1c} SE_{SAT}^{1c}}{E_{SAT}^{2c} SE_{SAT}^{2c}}} (\text{Indicator}(SE_{SAT}^{2c} > 0) + \text{Indicator}(SE_{SAT}^{2c} < 0))$	ConstService
SAT_A	$A_{SAT}^q$	$E_{SAT}^{2c} - \frac{E_{SAT}^{1c} - E_{SAT}^{2c}}{a_{SAT}-1}$	ConstService
SAT_B	$B_{SAT}^q$	$\frac{E_{SAT}^{2c} SE_{SAT}^{2c} (a_{SAT}-1)^2 (\text{Indicator}(a_{SAT}>0) + \text{Indicator}(a_{SAT}<0))}{(E_{SAT}^{1c} - E_{SAT}^{2c})^2}$	ConstService
Se0	$S_{e0}$	$\frac{B_{SAT}^q (A_{SAT}^q - v_{f0})^2 \text{Indicator}(v_{f0} > A_{SAT}^q)}{-u_g + v_{f0} + 1}$	ConstService
vr0	$V_{r0}$	$v_{f0} (K_E + S_{e0})$	ConstService
vb0	$V_{b0}$	$\frac{V_{r0}}{K_A}$	ConstService
vref0	$V_{ref0}$	$V_{ref}$	PostInitService

## Discretes

Name	Symbol	Type	Info
SL	$SL$	LessThan	
LL_LT1	$LT_{LL}$	LessThan	
LL_LT2	$LT_{LL}$	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

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)

### 5.12.2 IEEEEX1

IEEEEX1 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	<i>bool</i>	
name		device name			
syn		Synchronous generator idx			mandatory
TR	$T_R$	Sensing time constant	0.010	<i>p.u.</i>	
TA	$T_A$	Lag time constant in anti-windup lag	0.040	<i>p.u.</i>	
TC	$T_C$	Lead time constant in lead-lag	1	<i>p.u.</i>	
TB	$T_B$	Lag time constant in lead-lag	1	<i>p.u.</i>	
TE	$T_E$	Exciter integrator time constant	0.800	<i>p.u.</i>	
TF1	$T_{F1}$	Feedback washout time constant	1	<i>p.u.</i>	non_zero
KF1	$K_{F1}$	Feedback washout gain	0.030	<i>p.u.</i>	
KA	$K_A$	Gain in anti-windup lag TF	40	<i>p.u.</i>	
KE	$K_E$	Gain added to saturation	1	<i>p.u.</i>	
VRMAX	$V_{RMAX}$	Maximum excitation limit	7.300	<i>p.u.</i>	
VRMIN	$V_{RMIN}$	Minimum excitation limit	-7.300	<i>p.u.</i>	
E1	$E_1$	First saturation point	0	<i>p.u.</i>	
SE1	$S_{E1}$	Value at first saturation point	0	<i>p.u.</i>	
E2	$E_2$	Second saturation point	1	<i>p.u.</i>	
SE2	$S_{E2}$	Value at second saturation point	1	<i>p.u.</i>	
ug	$u_g$	Generator online status	0	<i>bool</i>	
Sn	$S_m$	Rated power from generator	0	<i>MVA</i>	
Vn	$V_m$	Rated voltage from generator	0	<i>kV</i>	
bus	$bus$	Bus idx of the generators	0		

## Variables

Name	Symbol	Type	Description	Unit	Properties
vp	$V_p$	State	Voltage after saturation feedback, before speed term	p.u.	v_str
LS_y	$y_{LS}$	State	State in lag transfer function		v_str
LL_x	$x'_{LL}$	State	State in lead-lag		v_str
LA_y	$y_{LA}$	State	State in lag TF		v_str
W_x	$x'_W$	State	State in washout filter		v_str
omega	$\omega$	ExtState	Generator speed		
v	$E_{term}$	Algeb	Input to exciter (bus v or Eterm)		v_str
vout	$v_{out}$	Algeb	Exciter final output voltage		v_str
vref	$V_{ref}$	Algeb	Reference voltage input	p.u.	v_str
Se	$S_e( V_{out} )$	Algeb	saturation output		v_str
vi	$V_i$	Algeb	Total input voltages	p.u.	v_str
LL_y	$y_{LL}$	Algeb	Output of lead-lag		v_str
W_y	$y_W$	Algeb	Output of washout filter		v_str
vf	$v_f$	ExtAlgeb	Excitation field voltage to generator		
Xad-Ifd	$X_{ad}I_{fd}$	ExtAlgeb	Armature excitation current		
a	$\theta$	ExtAlgeb	Bus voltage phase angle		
vbus	$V$	ExtAlgeb	Bus voltage magnitude		

## Initialization Equations

Name	Symbol	Type	Initial Value
vp	$V_p$	State	$v_{f0}$
LS_y	$y_{LS}$	State	$1.0E_{term}$
LL_x	$x'_{LL}$	State	$V_i$
LA_y	$y_{LA}$	State	$K_A y_{LL}$
W_x	$x'_W$	State	$V_p$
omega	$\omega$	ExtState	
v	$E_{term}$	Algeb	$V$
vout	$v_{out}$	Algeb	$u_e v_{f0}$
vref	$V_{ref}$	Algeb	$E_{term} + V_{b0}$
Se	$S_e( V_{out} )$	Algeb	$S_{e0}$
vi	$V_i$	Algeb	$V_{b0}$
LL_y	$y_{LL}$	Algeb	$V_i$
W_y	$y_W$	Algeb	0
vf	$v_f$	ExtAlgeb	
XadIfd	$X_{ad} I_{fd}$	ExtAlgeb	
a	$\theta$	ExtAlgeb	
vbus	$V$	ExtAlgeb	

## Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
vp	$V_p$	State	$u_e (-K_E V_p - S_e( V_{out} ) V_p + y_{LA})$	$T_E$
LS_y	$y_{LS}$	State	$1.0E_{term} - y_{LS}$	$T_R$
LL_x	$x'_{LL}$	State	$V_i - x'_{LL}$	$T_B$
LA_y	$y_{LA}$	State	$K_A y_{LL} - y_{LA}$	$T_A$
W_x	$x'_W$	State	$V_p - x'_W$	$T_{F1}$
omega	$\omega$	ExtState	0	

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
v	$E_{term}$	Algeb	$-E_{term} + V$
vout	$v_{out}$	Algeb	$V_p u_e - v_{out}$
vref	$V_{ref}$	Algeb	$V_{ref0} - V_{ref}$
Se	$S_e( V_{out} )$	Algeb	$B_{SAT}^q z_0^{SL} (-A_{SAT}^q + V_p)^2 - S_e( V_{out} ) V_p$
vi	$V_i$	Algeb	$-V_i + V_{ref} - y_{LS} - y_W$
LL_y	$y_{LL}$	Algeb	$T_B x'_{LL} - T_B y_{LL} + T_C (V_i - x'_{LL}) + (z_1^{LT_{LL}})^2 (-x'_{LL} + y_{LL})$
W_y	$y_W$	Algeb	$K_{F1} (V_p - x'_W) - T_{F1} y_W$
vf	$v_f$	ExtAlgeb	$u_e (-v_{f0} + v_{out})$
XadIfd	$X_{ad} I_{fd}$	ExtAlgeb	0
a	$\theta$	ExtAlgeb	0
vbus	$V$	ExtAlgeb	0

## Services

Name	Symbol	Equation	Type
ue	$u_e$	$uu_g$	ConstService
SAT_E1	$E_{SAT}^{1c}$	$E_1$	ConstService
SAT_E2	$E_{SAT}^{2c}$	$E_2 - 2z_{SAT}^{SE2} + 2$	ConstService
SAT_SE1	$SE_{SAT}^{1c}$	$S_E1$	ConstService
SAT_SE2	$SE_{SAT}^{2c}$	$S_E2 - 2z_{SAT}^{SE2} + 2$	ConstService
SAT_a	$a_{SAT}$	$\sqrt{\frac{E_{SAT}^{1c} SE_{SAT}^{1c}}{E_{SAT}^{2c} SE_{SAT}^{2c}}} (\text{Indicator}(SE_{SAT}^{2c} > 0) + \text{Indicator}(SE_{SAT}^{2c} <$	ConstService
SAT_A	$A_{SAT}^q$	$E_{SAT}^{2c} - \frac{E_{SAT}^{1c} - E_{SAT}^{2c}}{a_{SAT} - 1}$	ConstService
SAT_B	$B_{SAT}^q$	$\frac{E_{SAT}^{2c} SE_{SAT}^{2c} (a_{SAT} - 1)^2 (\text{Indicator}(a_{SAT} > 0) + \text{Indicator}(a_{SAT} < 0))}{(E_{SAT}^{1c} - E_{SAT}^{2c})^2}$	ConstService
Se0	$S_{e0}$	$\frac{B_{SAT}^q (A_{SAT}^q - v_{f0})^2 \text{Indicator}(v_{f0} > A_{SAT}^q)}{-u_g + v_{f0} + 1}$	ConstService
vr0	$V_{r0}$	$v_{f0} (K_E + S_{e0})$	ConstService
vb0	$V_{b0}$	$\frac{V_{r0}}{K_A}$	ConstService
vref0	$V_{ref0}$	$V_{ref}$	PostInitService
VRT-MAX	$V_{RMAX} V_T$	$E_{term} V_{RMAX}$	VarService
VRTMIN	$V_{RMIN} V_T$	$E_{term} V_{RMIN}$	VarService

## Discretes

Name	Symbol	Type	Info
SL	$SL$	LessThan	
LL_LT1	$LT_{LL}$	LessThan	
LL_LT2	$LT_{LL}$	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

Config Fields in [IEEEEX1]

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)

### 5.12.3 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	<i>bool</i>	
name		device name			
syn		Synchronous generator idx			mandatory
TR	$T_R$	Sensing time constant	0.010	<i>p.u.</i>	
KA	$K_A$	Regulator gain	80		
TA	$T_A$	Lag time constant in regulator	0.040	<i>p.u.</i>	
TB	$T_B$	Lag time constant in lead-lag	1	<i>p.u.</i>	
TC	$T_C$	Lead time constant in lead-lag	1	<i>p.u.</i>	
VRMAX	$V_{RMAX}$	Max. exc. limit (0-unlimited)	7.300	<i>p.u.</i>	
VRMIN	$V_{RMIN}$	Min. excitation limit	-7.300	<i>p.u.</i>	
KE	$K_E$	Saturation feedback gain	1	<i>p.u.</i>	
TE	$T_E$	Integrator time constant	0.800	<i>p.u.</i>	
KF	$K_F$	Feedback gain	0.100		
TF1	$T_{F1}$	Feedback washout time constant	1	<i>p.u.</i>	non_zero,non_negative
Switch	$S_w$	Switch that PSS/E did not implement	0	<i>bool</i>	
E1	$E_1$	First saturation point	0	<i>p.u.</i>	
SE1	$S_{E1}$	Value at first saturation point	0	<i>p.u.</i>	
E2	$E_2$	Second saturation point	0	<i>p.u.</i>	
SE2	$S_{E2}$	Value at second saturation point	0	<i>p.u.</i>	
ug	$u_g$	Generator online status	0	<i>bool</i>	
Sn	$S_m$	Rated power from generator	0	<i>MVA</i>	
Vn	$V_m$	Rated voltage from generator	0	<i>kV</i>	
bus	$bus$	Bus idx of the generators	0		

## Variables

Name	Symbol	Type	Description	Unit	Properties
LG_y	$y_{LG}$	State	State in lag transfer function	v_str	
LL_x	$x'_{LL}$	State	State in lead-lag	v_str	
LA_y	$y_{LA}$	State	State in lag TF	v_str	
INT_y	$y_{INT}$	State	Integrator output	v_str	
WF_x	$x'_{WF}$	State	State in washout filter	v_str	
omega	$\omega$	ExtState	Generator speed		
v	$E_{term}$	Algeb	Input to exciter (bus v or Eterm)	v_str	
vout	$v_{out}$	Algeb	Exciter final output voltage	v_str	
vref	$V_{ref}$	Algeb	Reference voltage input	p.u.	v_str
vi	$V_i$	Algeb	Total input voltages	p.u.	v_str
LL_y	$y_{LL}$	Algeb	Output of lead-lag	v_str	
UEL	$U_{EL}$	Algeb	Interface var for under exc. limiter	v_str	
HG_y	$y_{HG}$	Algeb	HVGate output	v_str	
Se	$V_{out} * S_e( V_{out} )$	Algeb	saturation output	v_str	
VFE	$V_{FE}$	Algeb	Combined saturation feedback	p.u.	v_str
WF_y	$y_{WF}$	Algeb	Output of washout filter	v_str	
vf	$v_f$	ExtAlgeb	Excitation field voltage to generator		
XadIfd	$X_{ad}I_{fd}$	ExtAlgeb	Armature excitation current		
a	$\theta$	ExtAlgeb	Bus voltage phase angle		
vbus	$V$	ExtAlgeb	Bus voltage magnitude		

## Initialization Equations

Name	Symbol	Type	Initial Value
LG_y	$y_{LG}$	State	$E_{term}$
LL_x	$x'_{LL}$	State	$V_i$
LA_y	$y_{LA}$	State	$K_A y_{LL}$
INT_y	$y_{INT}$	State	$v_{f0}$
WF_x	$x'_{WF}$	State	$y_{INT}$
omega	$\omega$	ExtState	
v	$E_{term}$	Algeb	$V$
vout	$v_{out}$	Algeb	$u_e v_{f0}$
vref	$V_{ref}$	Algeb	$E_{term} + \frac{V_{FE0}}{K_A}$
vi	$V_i$	Algeb	$\frac{V_{FE0}}{K_A}$
LL_y	$y_{LL}$	Algeb	$V_i$
UEL	$U_{EL}$	Algeb	0
HG_y	$y_{HG}$	Algeb	$U_{EL} z_0^{None_{HG}} + y_{LL} z_1^{None_{HG}}$
Se	$V_{out} * S_e( V_{out} )$	Algeb	$S_{e0}$
VFE	$V_{FE}$	Algeb	$V_{FE0}$
WF_y	$y_{WF}$	Algeb	0
vf	$v_f$	ExtAlgeb	
XadIfd	$X_{ad} I_{fd}$	ExtAlgeb	
a	$\theta$	ExtAlgeb	
vbus	$V$	ExtAlgeb	

## Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
LG_y	$y_{LG}$	State	$E_{term} - y_{LG}$	$T_R$
LL_x	$x'_{LL}$	State	$V_i - x'_{LL}$	$T_B$
LA_y	$y_{LA}$	State	$K_A y_{LL} - y_{LA}$	$T_A$
INT_y	$y_{INT}$	State	$u_e (-V_{FE} + y_{LA})$	$T_E$
WF_x	$x'_{WF}$	State	$-x'_{WF} + y_{INT}$	$T_{F1}$
omega	$\omega$	ExtState	0	

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
v	$E_{term}$	Algeb	$-E_{term} + V$
vout	$v_{out}$	Algeb	$-v_{out} + y_{INT}$
vref	$V_{ref}$	Algeb	$V_{ref0} - V_{ref}$
vi	$V_i$	Algeb	$-E_{term} - V_i + V_{ref} - y_{WF}$
LL_y	$y_{LL}$	Algeb	$T_B x'_{LL} - T_B y_{LL} + T_C (V_i - x'_{LL}) + \left(z_1^{LT_{LL}}\right)^2 (-x'_{LL} + y_{LL})$
UEL	$U_{EL}$	Algeb	$-U_{EL}$
HG_y	$y_{HG}$	Algeb	$U_{EL} z_0^{None_{HG}} - y_{HG} + y_{LL} z_1^{None_{HG}}$
Se	$V_{out} * S_e( V_{out} )$	Algeb	$B_{SAT}^q z_0^{SL} \left(-A_{SAT}^q + y_{INT}\right)^2 - V_{out} * S_e( V_{out} )$
VFE	$V_{FE}$	Algeb	$K_E y_{INT} - V_{FE} + V_{out} * S_e( V_{out} )$
WF_y	$y_{WF}$	Algeb	$K_F (-x'_{WF} + y_{INT}) - T_{F1} y_{WF}$
vf	$v_f$	ExtAl- geb	$u_e (-v_{f0} + v_{out})$
Xad- Ifd	$X_{ad} I_{fd}$	ExtAl- geb	0
a	$\theta$	ExtAl- geb	0
vbus	$V$	ExtAl- geb	0

## Services

Name	Symbol	Equation	Type
ue	$u_e$	$uu_g$	ConstService
VR-MAXc	$VRMAXc$	$VRMAX - 999z_{VRMAX} + 999$	ConstService
SAT_E1	$E_{SAT}^{1c}$	$E_1$	ConstService
SAT_E2	$E_{SAT}^{2c}$	$E_2 - 2z_{SAT}^{SE2} + 2$	ConstService
SAT_SE1	$SE_{SAT}^{1c}$	$S_E1$	ConstService
SAT_SE2	$SE_{SAT}^{2c}$	$S_E2 - 2z_{SAT}^{SE2} + 2$	ConstService
SAT_a	$a_{SAT}$	$\sqrt{\frac{E_{SAT}^{1c}SE_{SAT}^{1c}}{E_{SAT}^{2c}SE_{SAT}^{2c}}} (\text{Indicator}(SE_{SAT}^{2c} > 0) + \text{Indicator}(SE_{SAT}^{2c} < 0))$	ConstService
SAT_A	$A_{SAT}^q$	$E_{SAT}^{2c} - \frac{E_{SAT}^{1c} - E_{SAT}^{2c}}{a_{SAT}-1}$	ConstService
SAT_B	$B_{SAT}^q$	$\frac{E_{SAT}^{2c}SE_{SAT}^{2c}(a_{SAT}-1)^2(\text{Indicator}(a_{SAT}>0)+\text{Indicator}(a_{SAT}<0))}{(E_{SAT}^{1c} - E_{SAT}^{2c})^2}$	ConstService
Se0	$S_{e0}$	$B_{SAT}^q (A_{SAT}^q - v_{f0})^2 \text{Indicator}(v_{f0} > A_{SAT}^q)$	ConstService
vfe0	$V_{FE0}$	$K_E v_{f0} + S_{e0}$	ConstService
vref0	$V_{ref0}$	$V_{ref}$	PostInitService
VRU	$VRMAX$	$VRMAXc$	ConstService
VRL	$VRMIN$	$VRMIN$	ConstService

## Discretes

Name	Symbol	Type	Info
LL_LT1	$LT_{LL}$	LessThan	
LL_LT2	$LT_{LL}$	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

## 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)

**5.12.4 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	<i>bool</i>	
name		device name			
syn		Synchronous generator idx			mandatory
TR	$T_R$	Sensing time constant	0.010	<i>p.u.</i>	
KA	$K_A$	Regulator gain	80		
TA	$T_A$	Lag time constant in regulator	0.040	<i>p.u.</i>	
TB	$T_B$	Lag time constant in lead-lag	1	<i>p.u.</i>	
TC	$T_C$	Lead time constant in lead-lag	1	<i>p.u.</i>	
VRMAX	$V_{RMAX}$	Max. exc. limit (0-unlimited)	7.300	<i>p.u.</i>	
VRMIN	$V_{RMIN}$	Min. excitation limit	-7.300	<i>p.u.</i>	
KE	$K_E$	Saturation feedback gain	1	<i>p.u.</i>	
TE	$T_E$	Integrator time constant	0.800	<i>p.u.</i>	
KF	$K_F$	Feedback gain	0.100		
TF1	$T_{F1}$	Feedback washout time constant	1	<i>p.u.</i>	non_zero,non_negative
Switch	$S_w$	Switch that PSS/E did not implement	0	<i>bool</i>	
E1	$E_1$	First saturation point	0	<i>p.u.</i>	
SE1	$S_{E1}$	Value at first saturation point	0	<i>p.u.</i>	
E2	$E_2$	Second saturation point	0	<i>p.u.</i>	
SE2	$S_{E2}$	Value at second saturation point	0	<i>p.u.</i>	
ug	$u_g$	Generator online status	0	<i>bool</i>	
Sn	$S_m$	Rated power from generator	0	<i>MVA</i>	
Vn	$V_m$	Rated voltage from generator	0	<i>kV</i>	
bus	$bus$	Bus idx of the generators	0		

## Variables

Name	Symbol	Type	Description	Unit	Properties
LG_y	$y_{LG}$	State	State in lag transfer function	v_str	
LL_x	$x'_{LL}$	State	State in lead-lag	v_str	
LA_y	$y_{LA}$	State	State in lag TF	v_str	
INT_y	$y_{INT}$	State	Integrator output	v_str	
WF_x	$x'_{WF}$	State	State in washout filter	v_str	
omega	$\omega$	ExtState	Generator speed		
v	$E_{term}$	Algeb	Input to exciter (bus v or Eterm)	v_str	
vout	$v_{out}$	Algeb	Exciter final output voltage	v_str	
vref	$V_{ref}$	Algeb	Reference voltage input	p.u.	v_str
vi	$V_i$	Algeb	Total input voltages	p.u.	v_str
LL_y	$y_{LL}$	Algeb	Output of lead-lag	v_str	
UEL	$U_{EL}$	Algeb	Interface var for under exc. limiter	v_str	
HG_y	$y_{HG}$	Algeb	HVGate output	v_str	
Se	$V_{out} * S_e( V_{out} )$	Algeb	saturation output	v_str	
VFE	$V_{FE}$	Algeb	Combined saturation feedback	p.u.	v_str
WF_y	$y_{WF}$	Algeb	Output of washout filter	v_str	
vf	$v_f$	ExtAlgeb	Excitation field voltage to generator		
XadIfd	$X_{ad}I_{fd}$	ExtAlgeb	Armature excitation current		
a	$\theta$	ExtAlgeb	Bus voltage phase angle		
vbus	$V$	ExtAlgeb	Bus voltage magnitude		

## Initialization Equations

Name	Symbol	Type	Initial Value
LG_y	$y_{LG}$	State	$E_{term}$
LL_x	$x'_{LL}$	State	$V_i$
LA_y	$y_{LA}$	State	$K_{AYLL}$
INT_y	$y_{INT}$	State	$v_{f0}$
WF_x	$x'_{WF}$	State	$y_{INT}$
omega	$\omega$	ExtState	
v	$E_{term}$	Algeb	$V$
vout	$v_{out}$	Algeb	$u_e v_{f0}$
vref	$V_{ref}$	Algeb	$E_{term} + \frac{V_{FE0}}{K_A}$
vi	$V_i$	Algeb	$\frac{V_{FE0}}{K_A}$
LL_y	$y_{LL}$	Algeb	$V_i$
UEL	$U_{EL}$	Algeb	0
HG_y	$y_{HG}$	Algeb	$U_{EL} z_0^{None_{HG}} + y_{LL} z_1^{None_{HG}}$
Se	$V_{out} * S_e( V_{out} )$	Algeb	$S_{e0}$
VFE	$V_{FE}$	Algeb	$V_{FE0}$
WF_y	$y_{WF}$	Algeb	0
vf	$v_f$	ExtAlgeb	
XadIfd	$X_{ad} I_{fd}$	ExtAlgeb	
a	$\theta$	ExtAlgeb	
vbus	$V$	ExtAlgeb	

## Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
LG_y	$y_{LG}$	State	$E_{term} - y_{LG}$	$T_R$
LL_x	$x'_{LL}$	State	$V_i - x'_{LL}$	$T_B$
LA_y	$y_{LA}$	State	$K_{AYLL} - y_{LA}$	$T_A$
INT_y	$y_{INT}$	State	$u_e (-V_{FE} + y_{LA})$	$T_E$
WF_x	$x'_{WF}$	State	$-x'_{WF} + y_{INT}$	$T_{F1}$
omega	$\omega$	ExtState	0	

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
v	$E_{term}$	Algeb	$-E_{term} + V$
vout	$v_{out}$	Algeb	$-v_{out} + y_{INT}$
vref	$V_{ref}$	Algeb	$V_{ref0} - V_{ref}$
vi	$V_i$	Algeb	$-E_{term} - V_i + V_{ref} - y_{WF}$
LL_y	$y_{LL}$	Algeb	$T_B x'_{LL} - T_B y_{LL} + T_C (V_i - x'_{LL}) + \left(z_1^{LT_{LL}}\right)^2 (-x'_{LL} + y_{LL})$
UEL	$U_{EL}$	Algeb	$-U_{EL}$
HG_y	$y_{HG}$	Algeb	$U_{EL} z_0^{None_{HG}} - y_{HG} + y_{LL} z_1^{None_{HG}}$
Se	$V_{out} * S_e( V_{out} )$	Algeb	$B_{SAT}^q z_0^{SL} \left(-A_{SAT}^q + y_{INT}\right)^2 - V_{out} * S_e( V_{out} )$
VFE	$V_{FE}$	Algeb	$K_E y_{INT} - V_{FE} + V_{out} * S_e( V_{out} )$
WF_y	$y_{WF}$	Algeb	$K_F (-x'_{WF} + y_{INT}) - T_{F1} y_{WF}$
vf	$v_f$	ExtAl- geb	$u_e (-v_{f0} + v_{out})$
Xad- Ifd	$X_{ad} I_{fd}$	ExtAl- geb	0
a	$\theta$	ExtAl- geb	0
vbus	$V$	ExtAl- geb	0

## Services

Name	Symbol	Equation	Type
ue	$u_e$	$uu_g$	ConstService
VR-MAXc	$VRMAXc$	$VRMAX - 999z_{VRMAX} + 999$	ConstService
SAT_E1	$E_{SAT}^{1c}$	$E_1$	ConstService
SAT_E2	$E_{SAT}^{2c}$	$E_2 - 2z_{SAT}^{SE2} + 2$	ConstService
SAT_SE1	$SE_{SAT}^{1c}$	$S_{E1}$	ConstService
SAT_SE2	$SE_{SAT}^{2c}$	$S_{E2} - 2z_{SAT}^{SE2} + 2$	ConstService
SAT_a	$a_{SAT}$	$\sqrt{\frac{E_{SAT}^{1c}SE_{SAT}^{1c}}{E_{SAT}^{2c}SE_{SAT}^{2c}}} (\text{Indicator}(SE_{SAT}^{2c} > 0) + \text{Indicator}(SE_{SAT}^{2c} < 0))$	ConstService
SAT_A	$A_{SAT}^q$	$E_{SAT}^{2c} - \frac{E_{SAT}^{1c} - E_{SAT}^{2c}}{a_{SAT}-1}$	ConstService
SAT_B	$B_{SAT}^q$	$\frac{E_{SAT}^{2c}SE_{SAT}^{2c}(a_{SAT}-1)^2(\text{Indicator}(a_{SAT}>0)+\text{Indicator}(a_{SAT}<0))}{(E_{SAT}^{1c} - E_{SAT}^{2c})^2}$	ConstService
Se0	$S_{e0}$	$B_{SAT}^q (A_{SAT}^q - v_{f0})^2 \text{Indicator}(v_{f0} > A_{SAT}^q)$	ConstService
vfe0	$V_{FE0}$	$K_E v_{f0} + S_{e0}$	ConstService
vref0	$V_{ref0}$	$V_{ref}$	PostInitService
VRU	$V_T V_{RMAX}$	$E_{term} VRMAXc$	VarService
VRL	$V_T V_{RMIN}$	$E_{term} V_{RMIN}$	VarService

## Discretes

Name	Symbol	Type	Info
LL_LT1	$LT_{LL}$	LessThan	
LL_LT2	$LT_{LL}$	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

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)

## 5.12.5 EXST1

EXST1-type static excitation system.

### Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	$u$	connection status	1	<i>bool</i>	
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
ug	$u_g$	Generator online status	0	<i>bool</i>	
Sn	$S_m$	Rated power from generator	0	<i>MVA</i>	
Vn	$V_m$	Rated voltage from generator	0	<i>kV</i>	
bus	$bus$	Bus idx of the generators	0		

## Variables

Name	Symbol	Type	Description	Unit	Properties
LG_y	$y_{LG}$	State	State in lag transfer function		v_str
LL_x	$x'_{LL}$	State	State in lead-lag		v_str
LR_y	$y_{LR}$	State	State in lag transfer function		v_str
WF_x	$x'_{WF}$	State	State in washout filter		v_str
omega	$\omega$	ExtState	Generator speed		
v	$E_{term}$	Algeb	Input to exciter (bus v or Eterm)		v_str
vout	$v_{out}$	Algeb	Exciter final output voltage		v_str
vref	$V_{ref}$	Algeb	Reference voltage input	p.u.	v_str
vi	$V_i$	Algeb	Total input voltages	p.u.	v_str
vl	$V_l$	Algeb	Input after limiter		v_str
LL_y	$y_{LL}$	Algeb	Output of lead-lag		v_str
WF_y	$y_{WF}$	Algeb	Output of washout filter		v_str
vfmax	$V_{fmax}$	Algeb	Upper bound of output limiter		v_str
vfmin	$V_{fmin}$	Algeb	Lower bound of output limiter		v_str
vf	$v_f$	ExtAlgeb	Excitation field voltage to generator		
XadIfd	$X_{ad}I_{fd}$	ExtAlgeb	Armature excitation current		
a	$\theta$	ExtAlgeb	Bus voltage phase angle		
vbus	$V$	ExtAlgeb	Bus voltage magnitude		

## Initialization Equations

Name	Symbol	Type	Initial Value
LG_y	$y_{LG}$	State	$E_{term}$
LL_x	$x'_{LL}$	State	$V_l$
LR_y	$y_{LR}$	State	$K_A Y_{LL}$
WF_x	$x'_{WF}$	State	$y_{LR}$
omega	$\omega$	ExtState	
v	$E_{term}$	Algeb	$V$
vout	$v_{out}$	Algeb	$u_e v_{f0}$
vref	$V_{ref}$	Algeb	$E_{term} + \frac{v_{f0}}{K_A}$
vi	$V_i$	Algeb	$\frac{v_{f0}}{K_A}$
vl	$V_l$	Algeb	$V_i z_i^{HLI} + V_{IMAX} z_u^{HLI} + V_{IMIN} z_l^{HLI}$
LL_y	$y_{LL}$	Algeb	$V_l$
WF_y	$y_{WF}$	Algeb	0
vfmax	$V_{fmax}$	Algeb	$-K_C X_{ad} I_{fd} + V_{RMAX}$
vfmin	$V_{fmin}$	Algeb	$-K_C X_{ad} I_{fd} + V_{RMIN}$
vf	$v_f$	ExtAlgeb	
XadIfd	$X_{ad}I_{fd}$	ExtAlgeb	
a	$\theta$	ExtAlgeb	
vbus	$V$	ExtAlgeb	

## Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
LG_y	$y_{LG}$	State	$E_{term} - y_{LG}$	$T_R$
LL_x	$x'_{LL}$	State	$V_l - x'_{LL}$	$T_B$
LR_y	$y_{LR}$	State	$K_A y_{LL} - y_{LR}$	$T_A$
WF_x	$x'_{WF}$	State	$-x'_{WF} + y_{LR}$	$T_F$
omega	$\omega$	ExtState	0	

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
v	$E_{term}$	Algeb	$-E_{term} + V$
vout	$v_{out}$	Algeb	$u_e (V_{fmax} z_u^{HLR} + V_{fmin} z_l^{HLR} + y_{LR} z_i^{HLR}) - v_{out}$
vref	$V_{ref}$	Algeb	$V_{ref0} - V_{ref}$
vi	$V_i$	Algeb	$-V_i + V_{ref} - y_{LG} - y_{WF}$
vl	$V_l$	Algeb	$V_i z_i^{HLI} - V_l + V_{IMAX} z_u^{HLI} + V_{IMIN} z_l^{HLI}$
LL_y	$y_{LL}$	Algeb	$T_B x'_{LL} - T_B y_{LL} + T_C (V_l - x'_{LL}) + (z_1^{LT_{LL}})^2 (-x'_{LL} + y_{LL})$
WF_y	$y_{WF}$	Algeb	$K_F (-x'_{WF} + y_{LR}) - T_F y_{WF}$
vfmax	$V_{fmax}$	Algeb	$-K_C X_{ad} I_{fd} + V_{RMAX} - V_{fmax}$
vfmin	$V_{fmin}$	Algeb	$-K_C X_{ad} I_{fd} + V_{RMIN} - V_{fmin}$
vf	$v_f$	ExtAlgeb	$u_e (-v_{f0} + v_{out})$
XadIfd	$X_{ad} I_{fd}$	ExtAlgeb	0
a	$\theta$	ExtAlgeb	0
vbus	$V$	ExtAlgeb	0

## Services

Name	Symbol	Equation	Type
ue	$u_e$	$uu_g$	ConstService
vref0	$V_{ref0}$	$V_{ref}$	PostInitService

## Discretes

Name	Symbol	Type	Info
HLI	$HLI$	HardLimiter	Hard limiter on input
LL_LT1	$LT_{LL}$	LessThan	
LL_LT2	$LT_{LL}$	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	Stabilizing circuit feedback

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)

## 5.12.6 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		
VR-MAX	$V_{RMAX}$	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		
VB-MAX	$V_{BMAX}$	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		
VG-MAX	$V_{GMAX}$	VG upper limit	4	p.u.	
THETAP	$\theta_P$	Rectifier firing angle	0	degree	
TM	$K_C$	Inner field regulator forward time constant	0.100		
VM-MAX	$V_{MMAX}$	Maximum VM limit	1	p.u.	
VM-MIN	$V_{RMIN}$	Minimum VM limit	0.100	p.u.	
ug	$u_g$	Generator online status	0	bool	
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	$y_{LG}$	State	State in lag transfer function	v_str	
LL_x	$x'_{LL}$	State	State in lead-lag	v_str	
LAW1_y	$y_{LAW1}$	State	State in lag TF	v_str	
LAW2_y	$y_{LAW2}$	State	State in lag TF	v_str	
omega	$\omega$	ExtState	Generator speed		
v	$E_{term}$	Algeb	Input to exciter (bus v or Eterm)	v_str	
vout	$v_{out}$	Algeb	Exciter final output voltage	v_str	
UEL	$U_{EL}$	Algeb	Interface var for under exc. limiter	v_str	
IN	$I_N$	Algeb	Input to FEX	v_str	
FEX_y	$y_{FEX}$	Algeb	Output of piecewise	v_str	
VB_x	$x_{VB}$	Algeb	Value before limiter	v_str	
VB_y	$y_{VB}$	Algeb	Output after limiter and post gain	v_str	
VG_x	$x_{VG}$	Algeb	Value before limiter	v_str	
VG_y	$y_{VG}$	Algeb	Output after limiter and post gain	v_str	
vrs	$V_{RS}$	Algeb	VR subtract feedback VG	v_str	
vref	$V_{ref}$	Algeb	Reference voltage input	p.u.	v_str
vi	$V_i$	Algeb	Total input voltages	p.u.	v_str
vil	$V_{il}$	Algeb	Input voltage after limit	v_str	
HG_y	$y_{HG}$	Algeb	HVGate output	v_str	
LL_y	$y_{LL}$	Algeb	Output of lead-lag	v_str	
vf	$v_f$	ExtAlgeb	Excitation field voltage to generator		
XadIfd	$X_{ad}I_{fd}$	ExtAlgeb	Armature excitation current		
a	$\theta$	ExtAlgeb	Bus voltage phase angle		
vbus	$V$	ExtAlgeb	Bus voltage magnitude		
vd	$V_d$	ExtAlgeb	d-axis machine voltage		
vq	$V_q$	ExtAlgeb	q-axis machine voltage		
Id	$I_d$	ExtAlgeb	d-axis machine current		
Iq	$I_q$	ExtAlgeb	q-axis machine current		

## Initialization Equations

Name	Sym- bol	Type	Initial Value
LG_y	$y_{LG}$	State	$E_{term}$
LL_x	$x'_{LL}$	State	$y_{HG}$
LAW1	$y_{LAW}$	State	$K_A y_{LL}$
LAW2	$y_{LAW}$	State	$K_M V_{RS}$
omega	$\omega$	ExtStat	
v	$E_{term}$	Al- geb	$V$
vout	$v_{out}$	Al- geb	$u_e v_{f0}$
UEL	$U_{EL}$	Al- geb	$U_{EL0}$
IN	$I_N$	Al- geb	$\text{safe}_{div}(K_C X_{ad} I_{fd}, V_E)$
FEX_y	$y_{FEX}$	Al- geb	$\text{FixPiecewise}\left((1, I_N \leq 0), (1 - 0.577I_N, I_N \leq 0.433), \left(\sqrt{0.75 - I_N^2}, I_N \leq 0.75\right), (1.732 - 0.577I_N, I_N > 0.75)\right)$
VB_x	$x_{VB}$	Al- geb	$V_E y_{FEX}$
VB_y	$y_{VB}$	Al- geb	$V_{BMAX} z_u^{lim_{VB}} + x_{VB} z_i^{lim_{VB}}$
VG_x	$x_{VG}$	Al- geb	$K_G v_{out}$
VG_y	$y_{VG}$	Al- geb	$V_{GMAX} z_u^{lim_{VG}} + x_{VG} z_i^{lim_{VG}}$
vrs	$V_{RS}$	Al- geb	$\frac{\text{safe}_{div}(v_{f0}, y_{VB})}{K_M}$
vref	$V_{ref}$	Al- geb	$E_{term} + \frac{V_{RS} + y_{VG}}{K_A}$
vi	$V_i$	Al- geb	$-E_{term} + V_{ref}$
vil	$V_{il}$	Al- geb	$V_i z_i^{HLI} + V_{IMAX} z_u^{HLI} + V_{IMIN} z_l^{HLI}$
HG_y	$y_{HG}$	Al- geb	$U_{EL} z_0^{None_{HG}} + V_{il} z_1^{None_{HG}}$
LL_y	$y_{LL}$	Al- geb	$y_{HG}$
vf	$v_f$	Ex- tAl- geb	
Xad- Ifd	$X_{ad} I_f$	Ex- tAl- geb	
a	$\theta$	Ex- tAl- geb	
534		Ex- tAl- geb	
vbus	$V$	Ex- tAl- geb	

## Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
LG_y	$y_{LG}$	State	$E_{term} - y_{LG}$	$T_R$
LL_x	$x'_{LL}$	State	$-x'_{LL} + y_{HG}$	$T_B$
LAW1_y	$y_{LAW1}$	State	$K_A y_{LL} - y_{LAW1}$	$T_A$
LAW2_y	$y_{LAW2}$	State	$K_M V_{RS} - y_{LAW2}$	$K_C$
omega	$\omega$	ExtState	0	

## Algebraic Equations

Name	Sym- bol	Type	RHS of Equation "0 = g(x, y)"
v	$E_{term}$	Al- geb	$-E_{term} + V$
vout	$v_{out}$	Al- geb	$u_e y_{LAW2} y_{VB} - v_{out}$
UEL	$U_{EL}$	Al- geb	$U_{EL0} - U_{EL}$
IN	$I_N$	Al- geb	$u_e (-I_N V_E + K_C X_{ad} I_{fd})$
FEX_	$y_{FEX}$	Al- geb	$-y_{FEX} + \text{FixPiecewise} \left( (1, I_N \leq 0), (1 - 0.577 I_N, I_N \leq 0.433), \left( \sqrt{0.75 - I_N^2}, I_N \leq 0.75 \right), 0 \right)$
VB_x	$x_{VB}$	Al- geb	$V_E y_{FEX} - x_{VB}$
VB_y	$y_{VB}$	Al- geb	$V_{BMAX} z_u^{lim_{VB}} + x_{VB} z_i^{lim_{VB}} - y_{VB}$
VG_x	$x_{VG}$	Al- geb	$K_G v_{out} - x_{VG}$
VG_y	$y_{VG}$	Al- geb	$V_{GMAX} z_u^{lim_{VG}} + x_{VG} z_i^{lim_{VG}} - y_{VG}$
vrs	$V_{RS}$	Al- geb	$-V_{RS} + y_{LAW1} - y_{VG}$
vref	$V_{ref}$	Al- geb	$V_{ref0} - V_{ref}$
vi	$V_i$	Al- geb	$-V_i + V_{ref} - y_{LG}$
vil	$V_{il}$	Al- geb	$V_i z_i^{HLI} + V_{IMAX} z_u^{HLI} + V_{IMIN} z_l^{HLI} - V_{il}$
HG_y	$y_{HG}$	Al- geb	$U_{EL} z_0^{None_{HG}} + V_{il} z_1^{None_{HG}} - y_{HG}$
LL_y	$y_{LL}$	Al- geb	$T_B x'_{LL} - T_B y_{LL} + T_C (-x'_{LL} + y_{HG}) + \left( z_1^{LT_{LL}} \right)^2 (-x'_{LL} + y_{LL})$
vf	$v_f$	Ex- tAl- geb	$u_e (-v_{f0} + v_{out})$
Xad- Ifd	$X_{ad} I_f$	Ex- tAl- geb	0
a	$\theta$	Ex- tAl- geb	0
vbus	$V$	Ex- tAl- geb	0
vd	$V_d$	Ex- tAl- geb	0
vq	$V_q$	Ex- tAl- geb	0

## Services

Name	Symbol	Equation	Type
ue	$u_e$	$uu_g$	ConstService
KPC	$K_{PC}$	$K_P e^{i \text{ radians}(\theta_P)}$	ConstService
UEL0	$U_{EL0}$	-9999	ConstService
VE	$V_E$	$ K_{PC}(V_d + iV_q) + i(I_d + iI_q)(K_I + K_{PC}X_L) $	VarService
vref0	$V_{ref0}$	$V_{ref}$	PostInitService

## Discretes

Name	Symbol	Type	Info
VB_lim	$lim_{VB}$	HardLimiter	
VG_lim	$lim_{VG}$	HardLimiter	
HG_lt	$None_{HG}$	LessThan	
LL_LT1	$LT_{LL}$	LessThan	
LL_LT2	$LT_{LL}$	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

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)

## 5.12.7 SEXS

Simplified Excitation System

### Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	$u$	connection status	1	<i>bool</i>	
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		
ug	$u_g$	Generator online status	0	<i>bool</i>	
Sn	$S_m$	Rated power from generator	0	<i>MVA</i>	
Vn	$V_m$	Rated voltage from generator	0	<i>kV</i>	
bus	$bus$	Bus idx of the generators	0		

### Variables

Name	Symbol	Type	Description	Unit	Properties
LL_x	$x'_{LL}$	State	State in lead-lag		v_str
LAW_y	$y_{LAW}$	State	State in lag TF		v_str
omega	$\omega$	ExtState	Generator speed		
v	$E_{term}$	Algeb	Input to exciter (bus v or Eterm)		v_str
vout	$v_{out}$	Algeb	Exciter final output voltage		v_str
vref	$V_{ref}$	Algeb	Reference voltage input	<i>p.u.</i>	v_str
vi	$V_i$	Algeb	Total input voltages	<i>p.u.</i>	v_str
LL_y	$y_{LL}$	Algeb	Output of lead-lag		v_str
vf	$v_f$	ExtAlgeb	Excitation field voltage to generator		
XadIfd	$X_{ad}I_{fd}$	ExtAlgeb	Armature excitation current		
a	$\theta$	ExtAlgeb	Bus voltage phase angle		
vbus	$V$	ExtAlgeb	Bus voltage magnitude		

## Initialization Equations

Name	Symbol	Type	Initial Value
LL_x	$x'_{LL}$	State	$V_i$
LAW_y	$y_{LAW}$	State	$Ky_{LL}$
omega	$\omega$	ExtState	
v	$E_{term}$	Algeb	$V$
vout	$v_{out}$	Algeb	$u_e v_{f0}$
vref	$V_{ref}$	Algeb	$E_{term} + \frac{v_{f0}}{K}$
vi	$V_i$	Algeb	$\frac{v_{f0}}{K}$
LL_y	$y_{LL}$	Algeb	$V_i$
vf	$v_f$	ExtAlgeb	
XadIfd	$X_{ad}I_{fd}$	ExtAlgeb	
a	$\theta$	ExtAlgeb	
vbus	$V$	ExtAlgeb	

## Differential Equations

Name	Symbol	Type	RHS of Equation "T' x' = f(x, y)"	T (LHS)
LL_x	$x'_{LL}$	State	$V_i - x'_{LL}$	$T_B$
LAW_y	$y_{LAW}$	State	$Ky_{LL} - y_{LAW}$	$T_E$
omega	$\omega$	ExtState	0	

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
v	$E_{term}$	Algeb	$-E_{term} + V$
vout	$v_{out}$	Algeb	$u_e y_{LAW} - v_{out}$
vref	$V_{ref}$	Algeb	$V_{ref0} - V_{ref}$
vi	$V_i$	Algeb	$-E_{term} - V_i + V_{ref}$
LL_y	$y_{LL}$	Algeb	$TA(V_i - x'_{LL}) + T_B x'_{LL} - T_B y_{LL} + \left(z_1^{LT_{LL}}\right)^2 (-x'_{LL} + y_{LL})$
vf	$v_f$	ExtAlgeb	$u_e (-v_{f0} + v_{out})$
XadIfd	$X_{ad}I_{fd}$	ExtAlgeb	0
a	$\theta$	ExtAlgeb	0
vbus	$V$	ExtAlgeb	0

## Services

Name	Symbol	Equation	Type
ue	$u_e$	$uu_g$	ConstService
TA	$TA$	$T_A/T_B T_B$	ConstService
vref0	$V_{ref0}$	$V_{ref}$	PostInitService

## Discretes

Name	Symbol	Type	Info
LL_LT1	$LT_{LL}$	LessThan	
LL_LT2	$LT_{LL}$	LessThan	
LAW_lim	$lim_{LAW}$	AntiWindup	Limiter in Lag

## Blocks

Name	Symbol	Type	Info
LL	$LL$	LeadLag	
LAW	$LAW$	LagAntiWindup	

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)

## 5.12.8 IEEET1

IEEET1 exciter.

## Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	$u$	connection status	1	<i>bool</i>	
name		device name			
syn		Synchronous generator idx			mandatory
TR	$T_R$	Sensing time constant	0.020	<i>p.u.</i>	
KA	$K_A$	Regulator gain	5	<i>p.u.</i>	
TA	$T_A$	Lag time constant in anti-windup lag	0.040	<i>p.u.</i>	
VRMAX	$V_{RMAX}$	Maximum excitation limit	7.300	<i>p.u.</i>	
VRMIN	$V_{RMIN}$	Minimum excitation limit	-7.300	<i>p.u.</i>	
KE	$K_E$	Gain added to saturation	1	<i>p.u.</i>	
TE	$T_E$	Exciter integrator time constant	0.800	<i>p.u.</i>	
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	<i>bool</i>	
E1	$E_1$	First saturation point	0	<i>p.u.</i>	
SE1	$S_{E1}$	Value at first saturation point	0	<i>p.u.</i>	
E2	$E_2$	Second saturation point	1	<i>p.u.</i>	
SE2	$S_{E2}$	Value at second saturation point	1	<i>p.u.</i>	
ug	$u_g$	Generator online status	0	<i>bool</i>	
Sn	$S_m$	Rated power from generator	0	<i>MVA</i>	
Vn	$V_m$	Rated voltage from generator	0	<i>kV</i>	
bus	$bus$	Bus idx of the generators	0		

## Variables

Name	Symbol	Type	Description	Unit	Properties
LG_y	$y_{LG}$	State	State in lag transfer function		v_str
LA_y	$y_{LA}$	State	State in lag TF		v_str
INT_y	$y_{INT}$	State	Integrator output		v_str
WF_x	$x'_{WF}$	State	State in washout filter		v_str
omega	$\omega$	ExtState	Generator speed		
v	$E_{term}$	Algeb	Input to exciter (bus v or Eterm)		v_str
vout	$v_{out}$	Algeb	Exciter final output voltage		v_str
vref	$V_{ref}$	Algeb	Reference voltage input	p.u.	v_str
vi	$V_i$	Algeb	Total input voltages	p.u.	v_str
VFE	$V_{FE}$	Algeb	Combined saturation feedback	p.u.	v_str
Se	$S_e( V_{out} )$	Algeb	saturation output		v_str
WF_y	$y_{WF}$	Algeb	Output of washout filter		v_str
vf	$v_f$	ExtAlgeb	Excitation field voltage to generator		
XadIfd	$X_{ad}I_{fd}$	ExtAlgeb	Armature excitation current		
a	$\theta$	ExtAlgeb	Bus voltage phase angle		
vbus	$V$	ExtAlgeb	Bus voltage magnitude		

## Initialization Equations

Name	Symbol	Type	Initial Value
LG_y	$y_{LG}$	State	$E_{term}$
LA_y	$y_{LA}$	State	$K_A u_e (V_i - y_{WF})$
INT_y	$y_{INT}$	State	$v_{f0}$
WF_x	$x'_{WF}$	State	$v_{out}$
omega	$\omega$	ExtState	
v	$E_{term}$	Algeb	$V$
vout	$v_{out}$	Algeb	$u_e v_{f0}$
vref	$V_{ref}$	Algeb	$E_{term} + V_{b0}$
vi	$V_i$	Algeb	$-E_{term} + V_{ref}$
VFE	$V_{FE}$	Algeb	$V_{FE0}$
Se	$S_e( V_{out} )$	Algeb	$S_{e0}$
WF_y	$y_{WF}$	Algeb	0
vf	$v_f$	ExtAlgeb	
XadIfd	$X_{ad}I_{fd}$	ExtAlgeb	
a	$\theta$	ExtAlgeb	
vbus	$V$	ExtAlgeb	

## Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
LG_y	$y_{LG}$	State	$E_{term} - y_{LG}$	$T_R$
LA_y	$y_{LA}$	State	$K_A u_e (V_i - y_{WF}) - y_{LA}$	$T_A$
INT_y	$y_{INT}$	State	$u_e (-V_{FE} + y_{LA})$	$T_E$
WF_x	$x'_{WF}$	State	$v_{out} - x'_{WF}$	$T_F$
omega	$\omega$	ExtState	0	

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
v	$E_{term}$	Algeb	$-E_{term} + V$
vout	$v_{out}$	Algeb	$u_e (-v_{out} + y_{INT})$
vref	$V_{ref}$	Algeb	$V_{ref0} - V_{ref}$
vi	$V_i$	Algeb	$u_e (-V_i + V_{ref} - y_{LG})$
VFE	$V_{FE}$	Algeb	$u_e (K_E y_{INT} + S_e( V_{out} ) - V_{FE})$
Se	$S_e( V_{out} )$	Algeb	$B_{SAT}^q z_0^{SL} (-A_{SAT}^q + y_{INT})^2 - S_e( V_{out} )$
WF_y	$y_{WF}$	Algeb	$K_F (v_{out} - x'_{WF}) - T_F y_{WF}$
vf	$v_f$	ExtAlgeb	$u_e (-v_{f0} + v_{out})$
XadIfd	$X_{ad} I_{fd}$	ExtAlgeb	0
a	$\theta$	ExtAlgeb	0
vbus	$V$	ExtAlgeb	0

## Services

Name	Symbol	Equation	Type
ue	$u_e$	$uu_g$	ConstService
VR-MAXc	$VRMAXc$	$VRMAX - 999z_{VRMAX} + 999$	ConstService
SAT_E1	$E_{SAT}^{1c}$	$E_1$	ConstService
SAT_E2	$E_{SAT}^{2c}$	$E_2 - 2z_{SAT}^{SE2} + 2$	ConstService
SAT_SE1	$SE_{SAT}^{1c}$	$S_{E1}$	ConstService
SAT_SE2	$SE_{SAT}^{2c}$	$S_{E2} - 2z_{SAT}^{SE2} + 2$	ConstService
SAT_a	$a_{SAT}$	$\sqrt{\frac{E_{SAT}^{1c}SE_{SAT}^{1c}}{E_{SAT}^{2c}SE_{SAT}^{2c}}} (\text{Indicator}(SE_{SAT}^{2c} > 0) + \text{Indicator}(SE_{SAT}^{2c} < 0))$	ConstService
SAT_A	$A_{SAT}^q$	$E_{SAT}^{2c} - \frac{E_{SAT}^{1c} - E_{SAT}^{2c}}{a_{SAT} - 1}$	ConstService
SAT_B	$B_{SAT}^q$	$\frac{E_{SAT}^{2c}SE_{SAT}^{2c}(a_{SAT}-1)^2(\text{Indicator}(a_{SAT}>0)+\text{Indicator}(a_{SAT}<0))}{(E_{SAT}^{1c}-E_{SAT}^{2c})^2}$	ConstService
Se0	$S_{e0}$	$B_{SAT}^q (A_{SAT}^q - v_{f0})^2 \text{Indicator}(v_{f0} > A_{SAT}^q)$	ConstService
vr0	$V_{r0}$	$K_E v_{f0} + S_{e0}$	ConstService
vb0	$V_{b0}$	$\frac{V_{r0}}{K_A}$	ConstService
vfe0	$V_{FE0}$	$K_E v_{f0} + S_{e0}$	ConstService
vref0	$V_{ref0}$	$V_{ref}$	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

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)

## 5.12.9 EXAC1

EXAC1 model.

Ref: [https://www.powerworld.com/WebHelp/Content/TransientModels\\_HTML/Exciter%20EXAC1.htm](https://www.powerworld.com/WebHelp/Content/TransientModels_HTML/Exciter%20EXAC1.htm)

### Parameters

Name	Symbol	Description	De- fault	Unit	Properties
idx		unique device idx			
u	$u$	connection status	1	<i>bool</i>	
name		device name			
syn		Synchronous generator idx			mandatory
TR	$T_R$	Sensing time constant	0.010	<i>p.u.</i>	
TB	$T_B$	Lag time constant in lead-lag	1	<i>p.u.</i>	
TC	$T_C$	Lead time constant in lead-lag	1	<i>p.u.</i>	
KA	$K_A$	Regulator gain	80		
TA	$T_A$	Lag time constant in regulator	0.040	<i>p.u.</i>	
VR-MAX	$V_{RMAX}$	Maximum regulator output	8	<i>p.u.</i>	
VR-MIN	$V_{RMIN}$	Minimum regulator output	0	<i>p.u.</i>	
TE	$T_E$	Exciter integrator time constant	0.800	<i>p.u.</i>	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	<i>p.u.</i>	
E1	$E_1$	First saturation point	0	<i>p.u.</i>	
SE1	$S_{E1}$	Value at first saturation point	0	<i>p.u.</i>	
E2	$E_2$	Second saturation point	1	<i>p.u.</i>	
SE2	$S_{E2}$	Value at second saturation point	1	<i>p.u.</i>	
ug	$u_g$	Generator online status	0	<i>bool</i>	
Sn	$S_m$	Rated power from generator	0	<i>MVA</i>	
Vn	$V_m$	Rated voltage from generator	0	<i>kV</i>	
bus	$bus$	Bus idx of the generators	0		

## Variables

Name	Symbol	Type	Description	Unit	Properties
LG_y	$y_{LG}$	State	State in lag transfer function	v_str	
LL_x	$x'_{LL}$	State	State in lead-lag	v_str	
LA_y	$y_{LA}$	State	State in lag TF	v_str	
INT_y	$y_{INT}$	State	Integrator output	v_str,v_iter	
WF_x	$x'_{WF}$	State	State in washout filter	v_str	
omega	$\omega$	ExtState	Generator speed		
v	$E_{term}$	Algeb	Input to exciter (bus v or Eterm)	v_str	
vout	$v_{out}$	Algeb	Exciter final output voltage	v_str	
IN	$I_N$	Algeb	Input to FEX	v_str,v_iter	
FEX_y	$y_{FEX}$	Algeb	Output of piecewise	v_str,v_iter	
vi	$V_i$	Algeb	Total input voltages	p.u.	v_str
LL_y	$y_{LL}$	Algeb	Output of lead-lag	v_str	
Se	$V_{out} * S_e( V_{out} )$	Algeb	saturation output	v_str	
VFE	$V_{FE}$	Algeb	Combined saturation feedback	p.u.	v_str
vref	$V_{ref}$	Algeb	Reference voltage input	p.u.	v_str
WF_y	$y_{WF}$	Algeb	Output of washout filter	v_str	
vf	$v_f$	ExtAlgeb	Excitation field voltage to generator		
XadIfd	$X_{ad}I_{fd}$	ExtAlgeb	Armature excitation current		
a	$\theta$	ExtAlgeb	Bus voltage phase angle		
vbus	$V$	ExtAlgeb	Bus voltage magnitude		

## Initialization Equations

Nam	Symbol	Type	Initial Value
LG_	$y_{LG}$	State	$E_{term}$
LL_x	$x'_{LL}$	State	$V_i$
LA_	$y_{LA}$	State	$K_{AYLL}$
INT_	$y_{INT}$	State	$-v_{f0} + y_{FEXYINT}$
WF_	$x'_{WF}$	State	$V_{FE}$
omeg	$\omega$	ExtSta	
v	$E_{term}$	Al- geb	$V$
vout	$v_{out}$	Al- geb	$u_e v_{f0}$
IN	$I_N$	Al- geb	$-I_N y_{INT} + K_C X_{ad} I_{fd}$
FEX_	$y_{FEX}$	Al- geb	$-y_{FEX} + \text{FixPiecewise} \left( (1, I_N \leq 0), (1 - 0.577 I_N, I_N \leq 0.433), \left( \sqrt{0.75 - I_N^2}, I_N \leq 0.75 \right) \right)$
vi	$V_i$	Al- geb	$-E_{term} + V_{ref}$
LL_y	$y_{LL}$	Al- geb	$V_i$
Se	$V_{out} * S_e( V_{out} )$	Al- geb	$B_{SAT}^q (-A_{SAT}^q + y_{INT})^2 \text{Indicator}(y_{INT} > A_{SAT}^q)$
VFE	$V_{FE}$	Al- geb	$K_C X_{ad} I_{fd} + K_{EYINT} + V_{out} * S_e( V_{out} )$
vref	$V_{ref}$	Al- geb	$E_{term} + \frac{V_{FE}}{K_A}$
WF_	$y_{WF}$	Al- geb	0
vf	$v_f$	Ex- tAl- geb	
Xad- Ifd	$X_{ad} I_{fd}$	Ex- tAl- geb	
a	$\theta$	Ex- tAl- geb	
vbus	$V$	Ex- tAl- geb	

## Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
LG_y	$y_{LG}$	State	$E_{term} - y_{LG}$	$T_R$
LL_x	$x'_{LL}$	State	$V_i - x'_{LL}$	$T_B$
LA_y	$y_{LA}$	State	$K_A y_{LL} - y_{LA}$	$T_A$
INT_y	$y_{INT}$	State	$u_e (-V_{FE} + y_{LA})$	$T_E$
WF_x	$x'_{WF}$	State	$V_{FE} - x'_{WF}$	$T_F$
omega	$\omega$	ExtState	0	

## Algebraic Equations

Nam	Symbol	Type	RHS of Equation "0 = g(x, y)"
v	$E_{term}$	Al- geb	$-E_{term} + V$
vout	$v_{out}$	Al- geb	$u_e (-v_{out} + y_{FEX}y_{INT})$
IN	$I_N$	Al- geb	$u_e (-I_N y_{INT} + K_C X_{ad} I_{fd})$
FEX_	$y_{FEX}$	Al- geb	$-y_{FEX} + \text{FixPiecewise} \left( (1, I_N \leq 0), (1 - 0.577 I_N, I_N \leq 0.433), \left( \sqrt{0.75 - I_N^2}, I_N \leq 0.75 \right) \right)$
vi	$V_i$	Al- geb	$u_e (-E_{term} - V_i + V_{ref} - y_{WF})$
LL_y	$y_{LL}$	Al- geb	$T_B x'_{LL} - T_B y_{LL} + T_C (V_i - x'_{LL}) + \left( z_1^{LT_{LL}} \right)^2 (-x'_{LL} + y_{LL})$
Se	$V_{out} * S_e( V_{out} )$	Al- geb	$u_e \left( B_{SAT}^q z_0^{SL} (-A_{SAT}^q + y_{INT})^2 - V_{out} * S_e( V_{out} ) \right)$
VFE	$V_{FE}$	Al- geb	$u_e (K_C X_{ad} I_{fd} + K_E y_{INT} - V_{FE} + V_{out} * S_e( V_{out} ))$
vref	$V_{ref}$	Al- geb	$V_{ref0} - V_{ref}$
WF_	$y_{WF}$	Al- geb	$K_F (V_{FE} - x'_{WF}) - T_F y_{WF}$
vf	$v_f$	Ex- tAl- geb	$u_e (-v_{f0} + v_{out})$
Xad- Ifd	$X_{ad} I_{fd}$	Ex- tAl- geb	0
a	$\theta$	Ex- tAl- geb	0
vbus	$V$	Ex- tAl- geb	0

## Services

Name	Symbol	Equation	Type
ue	$u_e$	$uu_g$	ConstService
SAT_E1	$E_{SAT}^{1c}$	$E_1$	ConstService
SAT_E2	$E_{SAT}^{2c}$	$E_2 - 2z_{SAT}^{SE2} + 2$	ConstService
SAT_SE1	$SE_{SAT}^{1c}$	$S_E1$	ConstService
SAT_SE2	$SE_{SAT}^{2c}$	$SE_2 - 2z_{SAT}^{SE2} + 2$	ConstService
SAT_a	$a_{SAT}$	$\sqrt{\frac{E_{SAT}^{1c} SE_{SAT}^{1c}}{E_{SAT}^{2c} SE_{SAT}^{2c}}} (\text{Indicator}(SE_{SAT}^{2c} > 0) + \text{Indicator}(SE_{SAT}^{2c} < 0))$	ConstService
SAT_A	$A_{SAT}^q$	$E_{SAT}^{2c} - \frac{E_{SAT}^{1c} - E_{SAT}^{2c}}{a_{SAT} - 1}$	ConstService
SAT_B	$B_{SAT}^q$	$\frac{E_{SAT}^{2c} SE_{SAT}^{2c} (a_{SAT} - 1)^2 (\text{Indicator}(a_{SAT} > 0) + \text{Indicator}(a_{SAT} < 0))}{(E_{SAT}^{1c} - E_{SAT}^{2c})^2}$	ConstService
vref0	$V_{ref0}$	$V_{ref}$	PostInitService

## Discretes

Name	Symbol	Type	Info
LL_LT1	$LT_{LL}$	LessThan	
LL_LT2	$LT_{LL}$	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

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)

## 5.12.10 EXAC2

EXAC2 model.

Ref: [https://www.powerworld.com/WebHelp/Content/TransientModels\\_HTML/Exciter%20EXAC2.htm](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 constant 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		
TF	$T_F$	Feedback delay	1		non_zero,non_ne
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.	

continues on next

Table 14 – continued from previous page

Name	Symbol	Description	Default	Unit	Properties
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
ug	$u_g$	Generator online status	0	bool	
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	$y_{LG}$	State	State in lag transfer function	v_str	
LL_x	$x'_{LL}$	State	State in lead-lag	v_str	
LA_y	$y_{LA}$	State	State in lag TF	v_str	
INT_y	$y_{INT}$	State	Integrator output	v_str,v_iter	
WF_x	$x'_{WF}$	State	State in washout filter	v_str	
omega	$\omega$	ExtState	Generator speed		
v	$E_{term}$	Algeb	Input to exciter (bus v or Eterm)	v_str	
vout	$v_{out}$	Algeb	Exciter final output voltage	v_str	
IN	$I_N$	Algeb	Input to FEX	v_str,v_iter	
FEX_y	$y_{FEX}$	Algeb	Output of piecewise	v_str,v_iter	
vi	$V_i$	Algeb	Total input voltages	p.u.	v_str
LL_y	$y_{LL}$	Algeb	Output of lead-lag	v_str	
Se	$V_{out} * S_e( V_{out} )$	Algeb	saturation output	v_str	
VFE	$V_{FE}$	Algeb	Combined saturation feedback	p.u.	v_str
vref	$V_{ref}$	Algeb	Reference voltage input	p.u.	v_str
WF_y	$y_{WF}$	Algeb	Output of washout filter	v_str	
VHA	$V_{HA}$	Algeb		v_str	
VL	$V_L$	Algeb		v_str	
LVG_y	$y_{LVG}$	Algeb	LVGate output	v_str	
VR_x	$x_{VR}$	Algeb	Value before limiter	v_str	
VR_y	$y_{VR}$	Algeb	Output after limiter and post gain	v_str	
VLRx	$VLR_x$	Algeb		v_str	
vf	$v_f$	ExtAlgeb	Excitation field voltage to generator		
XadIfd	$X_{ad}I_{fd}$	ExtAlgeb	Armature excitation current		
a	$\theta$	ExtAlgeb	Bus voltage phase angle		
vbus	$V$	ExtAlgeb	Bus voltage magnitude		



## Initialization Equations

Name	Symbol	Type	Initial Value
LG_y	$y_{LG}$	State	$E_{term}$
LL_x	$x'_{LL}$	State	$V_i$
LA_y	$y_{LA}$	State	$K_A Y_{LL}$
INT_y	$y_{INT}$	State	$-v_{f0} + y_{FEX} y_{INT}$
WF_x	$x'_{WF}$	State	$V_{FE}$
omeg	$\omega$	ExtSta	
v	$E_{term}$	Algеб	$V$
vout	$v_{out}$	Algеб	$u_e v_{f0}$
IN	$I_N$	Algеб	$-I_N y_{INT} + K_C X_{ad} I_{fd}$
FEX_y	$y_{FEX}$	Algеб	$-y_{FEX} + \text{FixPiecewise} \left( (1, I_N \leq 0), (1 - 0.577 I_N, I_N \leq 0.433), \left( \sqrt{0.75 - I_N^2}, I_N \leq 0.75 \right) \right)$
vi	$V_i$	Algеб	$-E_{term} + V_{ref}$
LL_y	$y_{LL}$	Algеб	$V_i$
Se	$V_{out} * S_e( V_{out} )$	Algеб	$B_{SAT}^q (-A_{SAT}^q + y_{INT})^2 \text{Indicator}(y_{INT} > A_{SAT}^q)$
VFE	$V_{FE}$	Algеб	$K_C X_{ad} I_{fd} + K_E y_{INT} + V_{out} * S_e( V_{out} )$
vref	$V_{ref}$	Algеб	$E_{term} + \frac{K_L V_{FE} + \frac{V_{FE}}{K_B}}{K_A}$
WF_y	$y_{WF}$	Algеб	0
VHA	$VHA$	Algеб	$-K_H V_{FE} + y_{LA}$
VL	$VL$	Algеб	$K_L (VLRx - V_{FE})$
LVG_y	$y_{LVG}$	Algеб	$VHA z_1^{None_{LVG}} + VL z_0^{None_{LVG}}$
VR_x	$x_{VR}$	Algеб	$K_B y_{LVG}$
VR_y	$y_{VR}$	Algеб	$V_{MAX} z_u^{lim_{VR}} + V_{MIN} z_l^{lim_{VR}} + x_{VR} z_i^{lim_{VR}}$
VLR	$VLRx$	Algеб	$V_{LR} \text{Indicator} \left( V_{FE} + \frac{V_{FE}}{K_B K_L} < V_{LR} \right) + \left( V_{FE} + \frac{V_{FE}}{K_B K_L} \right) \text{Indicator} \left( V_{FE} + \frac{V_{FE}}{K_B K_L} > V_{LR} \right)$
vf	$v_f$	ExtAlgеб	
Xad-Ifd	$X_{ad} I_{fd}$	ExtAlgеб	
554_a	$\theta$	geb	
		ExtAlgеб	

## Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
LG_y	$y_{LG}$	State	$E_{term} - y_{LG}$	$T_R$
LL_x	$x'_{LL}$	State	$V_i - x'_{LL}$	$T_B$
LA_y	$y_{LA}$	State	$K_A y_{LL} - y_{LA}$	$T_A$
INT_y	$y_{INT}$	State	$u_e (-V_{FE} + y_{VR})$	$T_E$
WF_x	$x'_{WF}$	State	$V_{FE} - x'_{WF}$	$T_F$
omega	$\omega$	ExtState	0	

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
v	$E_{term}$	Algеб	$-E_{term} + V$
vout	$v_{out}$	Algеб	$u_e (-v_{out} + y_{FEX}y_{INT})$
IN	$I_N$	Algеб	$u_e (-I_N y_{INT} + K_C X_{ad} I_{fd})$
FEX_	$y_{FEX}$	Algеб	$-y_{FEX} + \text{FixPiecewise} \left( (1, I_N \leq 0), (1 - 0.577 I_N, I_N \leq 0.433), \left( \sqrt{0.75 - I_N^2}, I_N \leq 0.75 \right) \right)$
vi	$V_i$	Algеб	$u_e (-E_{term} - V_i + V_{ref} - y_{WF})$
LL_y	$y_{LL}$	Algеб	$T_B x'_{LL} - T_B y_{LL} + T_C (V_i - x'_{LL}) + \left( z_1^{LT_{LL}} \right)^2 (-x'_{LL} + y_{LL})$
Se	$V_{out} * S_e( V_{out} )$	Algеб	$u_e \left( B_{SAT}^q z_0^{SL} (-A_{SAT}^q + y_{INT})^2 - V_{out} * S_e( V_{out} ) \right)$
VFE	$V_{FE}$	Algеб	$u_e (K_C X_{ad} I_{fd} + K_E y_{INT} - V_{FE} + V_{out} * S_e( V_{out} ))$
vref	$V_{ref}$	Algеб	$V_{ref0} - V_{ref}$
WF_	$y_{WF}$	Algеб	$K_F (V_{FE} - x'_{WF}) - T_F y_{WF}$
VHA	$VHA$	Algеб	$-K_H V_{FE} - VHA + y_{LA}$
VL	$VL$	Algеб	$K_L (VLRx - V_{FE}) - VL$
LVG_	$y_{LVG}$	Algеб	$VHA z_1^{None_{LVG}} + VL z_0^{None_{LVG}} - y_{LVG}$
VR_x	$x_{VR}$	Algеб	$K_B y_{LVG} - x_{VR}$
VR_y	$y_{VR}$	Algеб	$V_{RMAX} z_u^{lim_{VR}} + V_{RMIN} z_l^{lim_{VR}} + x_{VR} z_i^{lim_{VR}} - y_{VR}$
VLR	$VLRx$	Algеб	$VLR_0 - VLRx$
vf	$v_f$	ExtAlgеб	$u_e (-v_{f0} + v_{out})$
Xad- Ifd	$X_{ad} I_{fd}$	ExtAlgеб	0
a	$\theta$	ExtAlgеб	0
vbus	$V$	ExtAlgеб	0

## Services

Name	Symbol	Equation	Type
ue	$u_e$	$uu_g$	ConstService
SAT_E1	$E_{SAT}^{1c}$	$E_1$	ConstService
SAT_E2	$E_{SAT}^{2c}$	$E_2 - 2z_{SAT}^{SE2} + 2$	ConstService
SAT_SE1	$SE_{SAT}^{1c}$	$S_E1$	ConstService
SAT_SE2	$SE_{SAT}^{2c}$	$SE_2 - 2z_{SAT}^{SE2} + 2$	ConstService
SAT_a	$a_{SAT}$	$\sqrt{\frac{E_{SAT}^{1c} SE_{SAT}^{1c}}{E_{SAT}^{2c} SE_{SAT}^{2c}}} (\text{Indicator}(SE_{SAT}^{2c} > 0) + \text{Indicator}(SE_{SAT}^{2c} < 0))$	ConstService
SAT_A	$A_{SAT}^q$	$E_{SAT}^{2c} - \frac{E_{SAT}^{1c} - E_{SAT}^{2c}}{a_{SAT} - 1}$	ConstService
SAT_B	$B_{SAT}^q$	$\frac{E_{SAT}^{2c} SE_{SAT}^{2c} (a_{SAT} - 1)^2 (\text{Indicator}(a_{SAT} > 0) + \text{Indicator}(a_{SAT} < 0))}{(E_{SAT}^{1c} - E_{SAT}^{2c})^2}$	ConstService
vref0	$V_{ref0}$	$V_{ref}$	PostInitService
VLR0	$VLR0$	$VLRx$	PostInitService

## Discretes

Name	Symbol	Type	Info
LL_LT1	$LT_{LL}$	LessThan	
LL_LT2	$LT_{LL}$	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	

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)

### 5.12.11 EXAC4

IEEE Type AC4 excitation system model.

#### Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	$u$	connection status	1	<i>bool</i>	
name		device name			
syn		Synchronous generator idx			mandatory
TR	$T_R$	Sensing time constant	0.010	<i>p.u.</i>	
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	<i>p.u.</i>	
TB	$T_B$	Lag time constant in lead-lag	1	<i>p.u.</i>	
KA	$K_A$	Regulator gain	80		
TA	$T_A$	Lag time constant in regulator	0.040	<i>p.u.</i>	
VRMAX	$V_{RMAX}$	Maximum excitation limit	8	<i>p.u.</i>	
VRMIN	$V_{RMIN}$	Minimum excitation limit	0	<i>p.u.</i>	
KC	$K_C$	Reactive power compensation gain	0		
ug	$u_g$	Generator online status	0	<i>bool</i>	
Sn	$S_m$	Rated power from generator	0	<i>MVA</i>	
Vn	$V_m$	Rated voltage from generator	0	<i>kV</i>	
bus	$bus$	Bus idx of the generators	0		

## Variables

Name	Symbol	Type	Description	Unit	Properties
LG_y	$y_{LG}$	State	State in lag transfer function		v_str
LL_x	$x'_{LL}$	State	State in lead-lag		v_str
LR_y	$y_{LR}$	State	State in lag transfer function		v_str
omega	$\omega$	ExtState	Generator speed		
v	$E_{term}$	Algeb	Input to exciter (bus v or Eterm)		v_str
vout	$v_{out}$	Algeb	Exciter final output voltage		v_str
vi	$V_i$	Algeb	Total input voltages	p.u.	v_str
LL_y	$y_{LL}$	Algeb	Output of lead-lag		v_str
vfmax	$V_{fmax}$	Algeb	Upper bound of output limiter		v_str
vfmin	$V_{fmin}$	Algeb	Lower bound of output limiter		v_str
vf	$v_f$	ExtAlgeb	Excitation field voltage to generator		
XadIfd	$X_{ad}I_{fd}$	ExtAlgeb	Armature excitation current		
a	$\theta$	ExtAlgeb	Bus voltage phase angle		
vbus	$V$	ExtAlgeb	Bus voltage magnitude		

## Initialization Equations

Name	Symbol	Type	Initial Value
LG_y	$y_{LG}$	State	$E_{term}$
LL_x	$x'_{LL}$	State	$V_i z_i^{HLI} + V_{IMAX} z_u^{HLI} + V_{IMIN} z_l^{HLI}$
LR_y	$y_{LR}$	State	$K_A y_{LL}$
omega	$\omega$	ExtState	
v	$E_{term}$	Algeb	$V$
vout	$v_{out}$	Algeb	$u_e v_{f0}$
vi	$V_i$	Algeb	$\frac{v_{f0}}{K_A}$
LL_y	$y_{LL}$	Algeb	$V_i z_i^{HLI} + V_{IMAX} z_u^{HLI} + V_{IMIN} z_l^{HLI}$
vfmax	$V_{fmax}$	Algeb	$-K_C X_{ad} I_{fd} + V_{RMAX}$
vfmin	$V_{fmin}$	Algeb	$-K_C X_{ad} I_{fd} + V_{RMIN}$
vf	$v_f$	ExtAlgeb	
XadIfd	$X_{ad}I_{fd}$	ExtAlgeb	
a	$\theta$	ExtAlgeb	
vbus	$V$	ExtAlgeb	

## Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
LG_y	$y_{LG}$	State	$E_{term} - y_{LG}$	$T_R$
LL_x	$x'_{LL}$	State	$V_i z_i^{HLI} + V_{IMAX} z_u^{HLI} + V_{IMIN} z_l^{HLI} - x'_{LL}$	$T_B$
LR_y	$y_{LR}$	State	$KAYLL - y_{LR}$	$T_A$
omega	$\omega$	ExtState	0	

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
v	$E_{term}$	Algeb	$-E_{term} + V$
vout	$v_{out}$	Algeb	$u_e (V_{fmax} z_u^{HLR} + V_{fmin} z_l^{HLR} + y_{LR} z_i^{HLR}) - v_{out}$
vi	$V_i$	Algeb	$-V_i + V_{ref0} - y_{LG}$
LL_y	$y_{LL}$	Algeb	$T_B x'_{LL} - T_B y_{LL} + T_C (V_i z_i^{HLI} + V_{IMAX} z_u^{HLI} + V_{IMIN} z_l^{HLI} - x'_{LL}) + (z_1^{LT_{LL}})^2 (-x'_{LL} + y_{LL})$
vf-	$V_{fmax}$	Algeb	$-K_C X_{ad} I_{fd} + V_{RMAX} - V_{fmax}$
max			
vfmin	$V_{fmin}$	Algeb	$-K_C X_{ad} I_{fd} + V_{RMIN} - V_{fmin}$
vf	$v_f$	ExtAl- geb	$u_e (-v_{f0} + v_{out})$
Xad-	$X_{ad} I_{fd}$	ExtAl- geb	0
Ifd			
a	$\theta$	ExtAl- geb	0
vbus	$V$	ExtAl- geb	0

## Services

Name	Symbol	Equation	Type
ue	$u_e$	$uu_g$	ConstService
vref0	$V_{ref0}$	$E_{term} + \frac{v_{f0}}{K_A}$	PostInitService

## Discretes

Name	Symbol	Type	Info
HLI	<i>HLI</i>	HardLimiter	Hard limiter on input
LL_LT1	<i>LT<sub>LL</sub></i>	LessThan	
LL_LT2	<i>LT<sub>LL</sub></i>	LessThan	
HLR	<i>HLR</i>	HardLimiter	Hard limiter on regulator output

## Blocks

Name	Symbol	Type	Info
LG	<i>LG</i>	Lag	Sensing delay
LL	<i>LL</i>	LeadLag	Lead-lag compensator
LR	<i>LR</i>	Lag	Regulator

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)

## 5.12.12 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.	
VR-MAX	$V_{RMAX}$	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.	
VM-MAX	$V_{RMAX}$	Maximum inner loop limit	8	p.u.	
VM-MIN	$V_{RMIN}$	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		
VB-MAX	$V_{BMAX}$	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	$\theta_P$	Rectifier firing angle	0	degree	
VG-MAX	$V_{GMAX}$	VG upper limit	20	p.u.	
ug	$u_g$	Generator online status	0	bool	
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	$y_{LG}$	State	State in lag transfer function		v_str
PI1_xi	$xi_{PI1}$	State	Integrator output		v_str
LA_y	$y_{LA}$	State	State in lag transfer function		v_str
PI2_xi	$xi_{PI2}$	State	Integrator output		v_str
omega	$\omega$	ExtState	Generator speed		
v	$E_{term}$	Algeb	Input to exciter (bus v or Eterm)		v_str
vout	$v_{out}$	Algeb	Exciter final output voltage		v_str
UEL	$U_{EL}$	Algeb	Interface var for under exc. limiter		v_str
IN	$I_N$	Algeb	Input to FEX		v_str
FEX_y	$y_{FEX}$	Algeb	Output of piecewise		v_str
VB_x	$x_{VB}$	Algeb	Value before limiter		v_str
VB_y	$y_{VB}$	Algeb	Output after limiter and post gain		v_str
VG_x	$x_{VG}$	Algeb	Value before limiter		v_str
VG_y	$y_{VG}$	Algeb	Output after limiter and post gain		v_str
vref	$V_{ref}$	Algeb	Reference voltage input	p.u.	v_str
vi	$V_i$	Algeb	Total input voltages	p.u.	v_str
PI1_ys	$y_{sPI1}$	Algeb	PI summation before limit		v_str
PI1_y	$y_{PI1}$	Algeb	PI output		v_str
PI2_ys	$y_{sPI2}$	Algeb	PI summation before limit		v_str
PI2_y	$y_{PI2}$	Algeb	PI output		v_str
vf	$v_f$	ExtAlgeb	Excitation field voltage to generator		
XadIfd	$X_{ad}I_{fd}$	ExtAlgeb	Armature excitation current		
a	$\theta$	ExtAlgeb	Bus voltage phase angle		
vbus	$V$	ExtAlgeb	Bus voltage magnitude		
vd	$V_d$	ExtAlgeb	d-axis machine voltage		
vq	$V_q$	ExtAlgeb	q-axis machine voltage		
Id	$I_d$	ExtAlgeb	d-axis machine current		
Iq	$I_q$	ExtAlgeb	q-axis machine current		

## Initialization Equations

Name	Symbol	Type	Initial Value
LG_y	$y_{LG}$	State	$E_{term}$
PI1_x	$x_{PI1}$	State	$y_{VG}$
LA_y	$y_{LA}$	State	$1.0y_{PI1}$
PI2_x	$x_{PI2}$	State	$\text{safe}_{div}(v_{f0}, y_{VB})$
omeg	$\omega$	ExtStat	
v	$E_{term}$	Al- geb	$V$
vout	$v_{out}$	Al- geb	$u_e v_{f0}$
UEL	$U_{EL}$	Al- geb	0
IN	$I_N$	Al- geb	$\text{safe}_{div}(K_C X_{ad} I_{fd}, V_E)$
FEX_y	$y_{FEX}$	Al- geb	$\text{FixPiecewise}\left((1, I_N \leq 0), (1 - 0.577I_N, I_N \leq 0.433), \left(\sqrt{0.75 - I_N^2}, I_N \leq 0.75\right), (1.732 - 0.577I_N, I_N > 0.75)\right)$
VB_x	$x_{VB}$	Al- geb	$V_E y_{FEX}$
VB_y	$y_{VB}$	Al- geb	$V_{BMAX} z_u^{lim_{VB}} + x_{VB} z_i^{lim_{VB}}$
VG_x	$x_{VG}$	Al- geb	$K_G v_{out}$
VG_y	$y_{VG}$	Al- geb	$V_{GMAX} z_u^{lim_{VG}} + x_{VG} z_i^{lim_{VG}}$
vref	$V_{ref}$	Al- geb	$E_{term}$
vi	$V_i$	Al- geb	$-E_{term} + V_{ref}$
PI1_y	$y_{SP1}$	Al- geb	$K_{PR} V_i + y_{VG}$
PI1_y	$y_{PI1}$	Al- geb	$V_{RMAX} z_u^{lim_{PI1}} + V_{RMIN} z_l^{lim_{PI1}} + y_{SP1} z_i^{lim_{PI1}}$
PI2_y	$y_{SP2}$	Al- geb	$K_{PM} (y_{LA} - y_{VG}) + \text{safe}_{div}(v_{f0}, y_{VB})$
PI2_y	$y_{PI2}$	Al- geb	$V_{RMAX} z_u^{lim_{PI2}} + V_{RMIN} z_l^{lim_{PI2}} + y_{SP2} z_i^{lim_{PI2}}$
vf	$v_f$	Ex- tAl- geb	
Xad- Ifd	$X_{ad} I_f$	Ex- tAl- geb	
a	$\theta$	Ex- tAl-	
564		geb	
vbus	$V$	Ex- tAl- geb	

## Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
LG_y	$y_{LG}$	State	$E_{term} - y_{LG}$	$T_R$
PI1_xi	$xi_{PI1}$	State	$K_{IR} (V_i + 2y_{PI1} - 2ys_{PI1})$	
LA_y	$y_{LA}$	State	$-y_{LA} + 1.0y_{PI1}$	$T_A$
PI2_xi	$xi_{PI2}$	State	$K_{IM} (y_{LA} + 2y_{PI2} - y_{VG} - 2ys_{PI2})$	
omega	$\omega$	ExtState	0	

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
v	$E_{term}$	Algеб	$-E_{term} + V$
vout	$v_{out}$	Algеб	$u_e y_{PI2} y_{VB} - v_{out}$
UEL	$U_{EL}$	Algеб	$-U_{EL}$
IN	$I_N$	Algеб	$u_e (-I_N V_E + K_C X_{ad} I_{fd})$
FEX_	$y_{FEX}$	Algеб	$-y_{FEX} + \text{FixPiecewise} \left( (1, I_N \leq 0), (1 - 0.577 I_N, I_N \leq 0.433), \left( \sqrt{0.75 - I_N^2}, I_N \leq 0.75 \right), 0 \right)$
VB_x	$x_{VB}$	Algеб	$V_E y_{FEX} - x_{VB}$
VB_y	$y_{VB}$	Algеб	$V_{BMAX} z_u^{lim_{VB}} + x_{VB} z_i^{lim_{VB}} - y_{VB}$
VG_x	$x_{VG}$	Algеб	$K_G v_{out} - x_{VG}$
VG_y	$y_{VG}$	Algеб	$V_{GMAX} z_u^{lim_{VG}} + x_{VG} z_i^{lim_{VG}} - y_{VG}$
vref	$V_{ref}$	Algеб	$V_{ref0} - V_{ref}$
vi	$V_i$	Algеб	$-V_i + V_{ref} - y_{LG}$
PI1_y	$y_{sPI1}$	Algеб	$K_{PR} V_i + x_{iPI1} - y_{sPI1}$
PI1_y	$y_{PI1}$	Algеб	$V_{RMAX} z_u^{lim_{PI1}} + V_{RMIN} z_l^{lim_{PI1}} - y_{PI1} + y_{sPI1} z_i^{lim_{PI1}}$
PI2_y	$y_{sPI2}$	Algеб	$K_{PM} (y_{LA} - y_{VG}) + x_{iPI2} - y_{sPI2}$
PI2_y	$y_{PI2}$	Algеб	$V_{RMAX} z_u^{lim_{PI2}} + V_{RMIN} z_l^{lim_{PI2}} - y_{PI2} + y_{sPI2} z_i^{lim_{PI2}}$
vf	$v_f$	ExtAlgеб	$u_e (-v_{f0} + v_{out})$
Xad-Ifd	$X_{ad} I_f$	ExtAlgеб	0
a	$\theta$	ExtAlgеб	0
vbus	$V$	ExtAlgеб	0
vd	$V_d$	ExtAlgеб	0
vq	$V_q$	ExtAlgеб	0

## Services

Name	Symbol	Equation	Type
ue	$u_e$	$uu_g$	ConstService
KPC	$K_{PC}$	$K_P e^{i \text{ radians}(\theta_P)}$	ConstService
VE	$V_E$	$ K_{PC}(V_d + iV_q) + i(I_d + iI_q)(K_I + K_{PC}X_L) $	VarService
vref0	$V_{ref0}$	$V_{ref}$	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	

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	

### 5.12.13 AC8B

Exciter AC8B model.

Reference:<sup>1</sup> <sup>2</sup>,

#### Parameters

Name	Symbol	Description	De-fault	Unit	Proper-ties
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.	
VR-MAX	$V_{RMAX}$	Maximum regulator limit	7.300	p.u.	
VRMIN	$V_{RMIN}$	Minimum regulator limit	1	p.u.	
VFE-MAX	$V_{FEMAX}$	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		
ug	$u_g$	Generator online status	0	bool	
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		

<sup>1</sup> Powerworld, Exciter AC8B, [Online], Available: [https://www.powerworld.com/WebHelp/Content/TransientModels\\_HTML/Exciter%20AC8B.htm](https://www.powerworld.com/WebHelp/Content/TransientModels_HTML/Exciter%20AC8B.htm)

<sup>2</sup> NEPLAN, Exciters Models, [Online], Available: [https://www.neplan.ch/wp-content/uploads/2015/08/Nep\\_EXCITERS1.pdf](https://www.neplan.ch/wp-content/uploads/2015/08/Nep_EXCITERS1.pdf)

## Variables

Name	Symbol	Type	Description	Unit	Properties
LG_y	$y_{LG}$	State	State in lag transfer function	v_str	
PID_xi	$x_{PID}$	State	Integrator output	v_str	
PID_WO_X	$x'_{PID WO_{PID}}$	State	State in washout filter	v_str	
LA_y	$y_{LA}$	State	State in lag TF	v_str	
INT_y	$y_{INT}$	State	Integrator output	v_str,v_iter	
omega	$\omega$	ExtState	Generator speed		
v	$E_{term}$	Algeb	Input to exciter (bus v or Eterm)	v_str	
vout	$v_{out}$	Algeb	Exciter final output voltage	v_str	
IN	$I_N$	Algeb	Input to FEX	v_str,v_iter	
FEX_y	$y_{FEX}$	Algeb	Output of piecewise	v_str,v_iter	
UEL	$U_{EL}$	Algeb	Interface var for under exc. limiter	v_str	
OEL	$O_{EL}$	Algeb	Interface var for over exc. limiter	v_str	
Vs	$V_s$	Algeb	Voltage compensation from PSS	v_str	
vref	$V_{ref}$	Algeb	Reference voltage input	p.u.	v_str
vi	$V_i$	Algeb	Total input voltages	p.u.	v_str
PID_uin	$uin_{PID}$	Algeb	PID input	v_str	
PID_WO_y	$y_{PID WO_{PID}}$	Algeb	Output of washout filter	v_str	
PID_ys	$y_{SPID}$	Algeb	PI summation before limit	v_str	
PID_y	$y_{PID}$	Algeb	PI output	v_str	
Se	$V_{out} * S_e( V_{out} )$	Algeb	saturation output	v_str	
VFE	$V_{FE}$	Algeb	Combined saturation feedback	p.u.	v_str
vf	$v_f$	ExtAlgeb	Excitation field voltage to generator		
XadIfd	$X_{ad}I_{fd}$	ExtAlgeb	Armature excitation current		
a	$\theta$	ExtAlgeb	Bus voltage phase angle		
vbus	$V$	ExtAlgeb	Bus voltage magnitude		

## Initialization Equations

Name	Symbol	Type	Initial Value
LG_y	$y_{LG}$	State	$E_{term}$
PID_xi	$x_{PID}$	State	$\frac{V_{FE}}{K_A}$
PID_W	$x'_{PID}$	State	$uin_{PID}$
LA_y	$y_{LA}$	State	$K_A y_{PID}$
INT_y	$y_{INT}$	State	$-v_{f0} + y_{FEX} y_{INT}$
omega	$\omega$	ExtSta	
v	$E_{term}$	Al- geb	$V$
vout	$v_{out}$	Al- geb	$u_e v_{f0}$
IN	$I_N$	Al- geb	$-I_N y_{INT} + K_C X_{ad} I_{fd}$
FEX_y	$y_{FEX}$	Al- geb	$-y_{FEX} + \text{FixPiecewise} \left( (1, I_N \leq 0), (1 - 0.577 I_N, I_N \leq 0.433), \left( \sqrt{0.75 - I_N^2}, I_N \leq 0.433 \right) \right)$
UEL	$U_{EL}$	Al- geb	$U_{EL_0}$
OEL	$O_{EL}$	Al- geb	$O_{EL_0}$
Vs	$V_s$	Al- geb	0
vref	$V_{ref}$	Al- geb	$E_{term}$
vi	$V_i$	Al- geb	$-E_{term} + V_{ref}$
PID_uir	$uin_{PID}$	Al- geb	$V_i$
PID_W	$y_{PID}$	Al- geb	0
PID_ys	$y_{SPID}$	Al- geb	$V_i k_P + \frac{V_{FE}}{K_A}$
PID_y	$y_{PID}$	Al- geb	$V_{P\text{MAX}} z_u^{lim_{PID}} + V_{P\text{MIN}} z_l^{lim_{PID}} + y_{SPID} z_i^{lim_{PID}}$
Se	$V_{out} * S_e( V_{out} )$	Al- geb	$B_{SAT}^q (-A_{SAT}^q + y_{INT})^2 \text{Indicator}(y_{INT} > A_{SAT}^q)$
VFE	$V_{FE}$	Al- geb	$K_D X_{ad} I_{fd} + K_E y_{INT} + V_{out} * S_e( V_{out} )$
vf	$v_f$	Ex- tAl- geb	
Xad- Ifd	$X_{ad} I_{fd}$	Ex- tAl- geb	
a	$\theta$	Ex- tAl- geb	
570		geb	
vbus	$V$	Ex- tAl- geb	

## Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
LG_y	$y_{LG}$	State	$E_{term} - y_{LG}$	$T_R$
PID_xi	$x_{PID}$	State	$k_I (V_i + 2y_{PID} - 2y_{SPID})$	
PID_WO_x	$x'_{PID WO_{PID}}$	State	$u_{inPID} - x'_{PID WO_{PID}}$	$T_d$
LA_y	$y_{LA}$	State	$K_{AYPID} - y_{LA}$	$T_A$
INT_y	$y_{INT}$	State	$u_e (-V_{FE} + y_{LA})$	$T_E$
omega	$\omega$	ExtState	0	

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
v	$E_{term}$	Al- geb	$-E_{term} + V$
vout	$v_{out}$	Al- geb	$u_e (-v_{out} + y_{FEXYINT})$
IN	$I_N$	Al- geb	$u_e (-I_N y_{INT} + K_C X_{ad} I_{fd})$
FEX_y	$y_{FEX}$	Al- geb	$-y_{FEX} + \text{FixPiecewise} \left( (1, I_N \leq 0), (1 - 0.577 I_N, I_N \leq 0.433), \left( \sqrt{0.75 - I_N^2}, I_N \leq 0.433 \right) \right)$
UEL	$U_{EL}$	Al- geb	$UEL_0 - U_{EL}$
OEL	$O_{EL}$	Al- geb	$OEL_0 - O_{EL}$
Vs	$V_s$	Al- geb	$-V_s$
vref	$V_{ref}$	Al- geb	$V_{ref0} - V_{ref}$
vi	$V_i$	Al- geb	$u_e (O_{EL} + U_{EL} - V_i + V_{ref} + V_s - y_{LG})$
PID_uir	$uinPID$	Al- geb	$V_i - uinPID$
PID_W	$y_{PID\ WO_F}$	Al- geb	$-T_d y_{PID\ WO_F} + k_D \left( uinPID - x'_{PID\ WO_F} \right)$
PID_ys	$y_{SPID}$	Al- geb	$V_i k_P + x_{PID} + y_{PID\ WO_F} - y_{SPID}$
PID_y	$y_{PID}$	Al- geb	$V_{PMAX} z_u^{lim_{PID}} + V_{PMIN} z_l^{lim_{PID}} - y_{PID} + y_{SPID} z_i^{lim_{PID}}$
Se	$V_{out} * S_e( V_{out} )$	Al- geb	$u_e \left( B_{SAT}^q z_0^{SL} (-A_{SAT}^q + y_{INT})^2 - V_{out} * S_e( V_{out} ) \right)$
VFE	$V_{FE}$	Al- geb	$u_e (K_D X_{ad} I_{fd} + K_E y_{INT} - V_{FE} + V_{out} * S_e( V_{out} ))$
vf	$v_f$	Ex- tAl- geb	$u_e (-v_{f0} + v_{out})$
Xad- Ifd	$X_{ad} I_{fd}$	Ex- tAl- geb	0
a	$\theta$	Ex- tAl- geb	0
vbus	$V$	Ex- tAl- geb	0

## Services

Name	Symbol	Equation	Type
ue	$u_e$	$uu_g$	ConstService
UEL0	$UEL0$	0	ConstService
OEL0	$OEL0$	0	ConstService
SAT_E1	$E_{SAT}^{1c}$	$E_1$	ConstService
SAT_E2	$E_{SAT}^{2c}$	$E_2 - 2z_{SAT}^{SE2} + 2$	ConstService
SAT_SE1	$SE_{SAT}^{1c}$	$S_E1$	ConstService
SAT_SE2	$SE_{SAT}^{2c}$	$S_E2 - 2z_{SAT}^{SE2} + 2$	ConstService
SAT_a	$a_{SAT}$	$\sqrt{\frac{E_{SAT}^{1c} SE_{SAT}^{1c}}{E_{SAT}^{2c} SE_{SAT}^{2c}}} (\text{Indicator } (SE_{SAT}^{2c} > 0) + \text{Indicator } (SE_{SAT}^{2c} < 0))$	ConstService
SAT_A	$A_{SAT}^q$	$E_{SAT}^{2c} - \frac{E_{SAT}^{1c} - E_{SAT}^{2c}}{a_{SAT} - 1}$	ConstService
SAT_B	$B_{SAT}^q$	$\frac{E_{SAT}^{2c} SE_{SAT}^{2c} (a_{SAT} - 1)^2 (\text{Indicator } (a_{SAT} > 0) + \text{Indicator } (a_{SAT} < 0))}{(E_{SAT}^{1c} - E_{SAT}^{2c})^2}$	ConstService
vref0	$V_{ref0}$	$E_{term}$	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

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	

### 5.12.14 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.powerworld.com/WebHelp/Content/TransientModels_HTML/Exciter%20IEEET3.htm)

[https://www.neplan.ch/wp-content/uploads/2015/08/Nep\\_EXCITERS1.pdf](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		
ug	$u_g$	Generator online status	0	bool	
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	$y_{LG}$	State	State in lag transfer function	v_str	
LA3_y	$y_{LA3}$	State	State in lag TF	v_str	
LA1_y	$y_{LA1}$	State	State in lag transfer function	v_str	
WF_x	$x'_{WF}$	State	State in washout filter	v_str	
omega	$\omega$	ExtState	Generator speed		
v	$E_{term}$	Algeb	Input to exciter (bus v or Eterm)	v_str	
vout	$v_{out}$	Algeb	Exciter final output voltage	v_str	
UEL	$U_{EL}$	Algeb	Interface var for under exc. limiter	v_str	
OEL	$O_{EL}$	Algeb	Interface var for over exc. limiter	v_str	
Vs	$V_s$	Algeb	Voltage compensation from PSS	v_str	
vref	$V_{ref}$	Algeb	Reference voltage input	p.u.	v_str
vi	$V_i$	Algeb	Total input voltages	p.u.	v_str
WF_y	$y_{WF}$	Algeb	Output of washout filter	v_str	
SQE	$SQE$	Algeb	Square of error after mul	v_str	
VB_y	$y_{VB}$	Algeb	Output of piecewise	v_str	
vf	$v_f$	ExtAlgeb	Excitation field voltage to generator		
XadIfd	$X_{ad}I_{fd}$	ExtAlgeb	Armature excitation current		
a	$\theta$	ExtAlgeb	Bus voltage phase angle		
vbus	$V$	ExtAlgeb	Bus voltage magnitude		
vd	$V_d$	ExtAlgeb	d-axis machine voltage		
vq	$V_q$	ExtAlgeb	q-axis machine voltage		
Id	$I_d$	ExtAlgeb	d-axis machine current		
Iq	$I_q$	ExtAlgeb	q-axis machine current		

## Initialization Equations

Name	Symbol	Type	Initial Value
LG_y	$y_{LG}$	State	$E_{term}$
LA3_y	$y_{LA3}$	State	$K_A u_e (V_i - y_{WF})$
LA1_y	$y_{LA1}$	State	$\frac{u_e (V_{BMAX} z_u^{HL} + y_{VB} z_i^{HL})}{K_E}$
WF_x	$x'_{WF}$	State	$y_{LA1}$
omega	$\omega$	ExtState	
v	$E_{term}$	Algeb	$V$
vout	$v_{out}$	Algeb	$u_e v_{f0}$
UEL	$U_{EL}$	Algeb	$UEL_0$
OEL	$O_{EL}$	Algeb	$OEL_0$
Vs	$V_s$	Algeb	0
vref	$V_{ref}$	Algeb	$E_{term} + V_{b0}$
vi	$V_i$	Algeb	$-E_{term} + V_{ref}$
WF_y	$y_{WF}$	Algeb	0
SQE	$SQE$	Algeb	$V_E^2 - 0.6084 X_{ad} I_{fd}^2$
VB_y	$y_{VB}$	Algeb	FixPiecewise (( $u_e y_{LA3}$ , $SQE \leq 0$ ), ( $u_e (\sqrt{SQE} + y_{LA3})$ , True))
vf	$v_f$	ExtAlgeb	
XadIfd	$X_{ad} I_{fd}$	ExtAlgeb	
a	$\theta$	ExtAlgeb	
vbus	$V$	ExtAlgeb	
vd	$V_d$	ExtAlgeb	
vq	$V_q$	ExtAlgeb	
Id	$I_d$	ExtAlgeb	
Iq	$I_q$	ExtAlgeb	

## Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
LG_y	$y_{LG}$	State	$E_{term} - y_{LG}$	$T_R$
LA3_y	$y_{LA3}$	State	$K_A u_e (V_i - y_{WF}) - y_{LA3}$	$T_A$
LA1_y	$y_{LA1}$	State	$-K_E y_{LA1} + u_e (V_{BMAX} z_u^{HL} + y_{VB} z_i^{HL})$	$T_E$
WF_x	$x'_{WF}$	State	$-x'_{WF} + y_{LA1}$	$T_F$
omega	$\omega$	ExtState	0	

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
v	$E_{term}$	Algeb	$-E_{term} + V$
vout	$v_{out}$	Algeb	$u_e (-v_{out} + y_{LA1})$
UEL	$UEL$	Algeb	$UEL_0 - U_{EL}$
OEL	$O_{EL}$	Algeb	$OEL_0 - O_{EL}$
Vs	$V_s$	Algeb	$-V_s$
vref	$V_{ref}$	Algeb	$V_{ref0} - V_{ref}$
vi	$V_i$	Algeb	$u_e (O_{EL} + U_{EL} - V_i + V_{ref} + V_s - y_{LG})$
WF_y	$y_{WF}$	Algeb	$K_F (-x'_{WF} + y_{LA1}) - T_F y_{WF}$
SQE	$SQE$	Algeb	$-SQE + V_E^2 - 0.6084X_{ad}I_{fd}^2$
VB_y	$y_{VB}$	Algeb	$-y_{VB} + \text{FixPiecewise} ((u_e y_{LA3}, SQE \leq 0), (u_e (\sqrt{SQE} + y_{LA3}), \text{True}))$
vf	$v_f$	ExtAlgeb	$u_e (-v_{f0} + v_{out})$
Xad-Ifd	$X_{ad}I_{fd}$	ExtAlgeb	0
a	$\theta$	ExtAlgeb	0
vbus	$V$	ExtAlgeb	0
vd	$V_d$	ExtAlgeb	0
vq	$V_q$	ExtAlgeb	0
Id	$I_d$	ExtAlgeb	0
Iq	$I_q$	ExtAlgeb	0

## Services

Name	Symbol	Equation	Type
ue	$u_e$	$uu_g$	ConstService
VE	$V_E$	$ iK_I(I_d + iI_q) + K_P(V_d + iV_q) $	VarService
V40	$V_{40}$	$\sqrt{V_E^2 - 0.6084X_{ad}I_{fd}^2}$	ConstService
VR0	$V_{R0}$	$K_E v_{f0} - V_{40}$	ConstService
vb0	$V_{b0}$	$\frac{V_{R0}}{K_A}$	ConstService
VRMAXc	$VRMAXc$	$VRMAX - 999z_{VRMAX} + 999$	ConstService
UEL0	$UEL0$	0	ConstService
OEL0	$OEL0$	0	ConstService
vref0	$V_{ref0}$	$V_{ref}$	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	$V_F$ , stabilizing circuit feedback, washout
VB	$VB$	Piecewise	

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)

### 5.12.15 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
VA-MAX	$V_{AMAX}$	$V_A$ 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
VR-MAX	$V_{RMAX}$	Max. exc. limit (0-unlimited)	7.300	p.u.	
VR-MIN	$V_{RMIN}$	Min. excitation limit	-	p.u.	
			7.300		
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	
ug	$u_g$	Generator online status	0	bool	
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	$y_{LG}$	State	State in lag transfer function	v_str	
LL_x	$x'_{LL}$	State	State in lead-lag	v_str	
LA_y	$y_{LA}$	State	State in lag TF	v_str	
INT_y	$y_{INT}$	State	Integrator output	v_str,v_iter	
WF_x	$x'_{WF}$	State	State in washout filter	v_str	
omega	$\omega$	ExtState	Generator speed		
v	$E_{term}$	Algeb	Input to exciter (bus v or Eterm)	v_str	
vout	$v_{out}$	Algeb	Exciter final output voltage	v_str	
UEL	$U_{EL}$	Algeb	Interface var for under exc. limiter	v_str	
OEL	$O_{EL}$	Algeb	Interface var for over exc. limiter	v_str	
Vs	$V_s$	Algeb	Voltage compensation from PSS	v_str	
vref	$V_{ref}$	Algeb	Reference voltage input	p.u.	v_str
IN	$I_N$	Algeb	Input to FEX	v_str,v_iter	
FEX_y	$y_{FEX}$	Algeb	Output of piecewise	v_str,v_iter	
vi	$V_i$	Algeb	Total input voltages	p.u.	v_str
LL_y	$y_{LL}$	Algeb	Output of lead-lag	v_str	
HVG_y	$y_{HVG}$	Algeb	HVGate output	v_str	
LVG_y	$y_{LVG}$	Algeb	LVGate output	v_str	
Se	$V_{out} * S_e( V_{out} )$	Algeb	saturation output	v_str	
VFE	$V_{FE}$	Algeb	Combined saturation feedback	p.u.	v_str
WF_y	$y_{WF}$	Algeb	Output of washout filter	v_str	
vf	$v_f$	ExtAlgeb	Excitation field voltage to generator		
XadIfd	$X_{ad}I_{fd}$	ExtAlgeb	Armature excitation current		
a	$\theta$	ExtAlgeb	Bus voltage phase angle		
vbus	$V$	ExtAlgeb	Bus voltage magnitude		

## Initialization Equations

Name	Symbol	Type	Initial Value
LG_y	$y_{LG}$	State	$E_{term}$
LL_x	$x'_{LL}$	State	$V_i$
LA_y	$y_{LA}$	State	$K_A Y_{LL}$
INT_	$y_{INT}$	State	$-v_{f0} + y_{FEX} y_{INT}$
WF_x	$x'_{WF}$	State	$V_{FE}$
omeg	$\omega$	ExtSta	
v	$E_{term}$	Algеб	$V$
vout	$v_{out}$	Algеб	$u_e v_{f0}$
UEL	$U_{EL}$	Algеб	$UEL_0$
OEL	$O_{EL}$	Algеб	$OEL_0$
Vs	$V_s$	Algеб	0
vref	$V_{ref}$	Algеб	$E_{term} + \frac{V_{FE}}{K_A}$
IN	$I_N$	Algеб	$-I_N y_{INT} + K_C X_{ad} I_{fd}$
FEX_	$y_{FEX}$	Algеб	$-y_{FEX} + \text{FixPiecewise} \left( (1, I_N \leq 0), (1 - 0.577 I_N, I_N \leq 0.433), \left( \sqrt{0.75 - I_N^2}, I_N \leq 0.75 \right) \right)$
vi	$V_i$	Algеб	$-E_{term} + V_{ref}$
LL_y	$y_{LL}$	Algеб	$V_i$
HVG_	$y_{HVG}$	Algеб	$U_{EL} z_0^{None_{HVG}} + y_{LA} z_1^{None_{HVG}}$
LVG_	$y_{LVG}$	Algеб	$O_{EL} z_0^{None_{LVG}} + y_{HVG} z_1^{None_{LVG}}$
Se	$V_{out} * S_e( V_{out} )$	Algеб	$B_{SAT}^q (-A_{SAT}^q + y_{INT})^2 \text{Indicator}(y_{INT} > A_{SAT}^q)$
VFE	$V_{FE}$	Algеб	$K_D X_{ad} I_{fd} + K_E y_{INT} + V_{out} * S_e( V_{out} )$
WF_y	$y_{WF}$	Algеб	0
vf	$v_f$	ExtAlgеб	
Xad- Ifd	$X_{ad} I_{fd}$	ExtAlgеб	
a	$\theta$	ExtAlgеб	
<b>5.12. Exciter</b>	geb		<b>581</b>
vbus	$V$	ExtAlgеб	

## Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
LG_y	$y_{LG}$	State	$E_{term} - y_{LG}$	$T_R$
LL_x	$x'_{LL}$	State	$V_i - x'_{LL}$	$T_B$
LA_y	$y_{LA}$	State	$K_A y_{LL} - y_{LA}$	$T_A$
INT_y	$y_{INT}$	State	$u_e (-V_{FE} + y_{LVG})$	$T_E$
WF_x	$x'_{WF}$	State	$V_{FE} - x'_{WF}$	$T_{F1}$
omega	$\omega$	ExtState	0	

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
v	$E_{term}$	Algеб	$-E_{term} + V$
vout	$v_{out}$	Algеб	$u_e y_{FEX} y_{INT} - v_{out}$
UEL	$U_{EL}$	Algеб	$UEL_0 - U_{EL}$
OEL	$O_{EL}$	Algеб	$OEL_0 - O_{EL}$
Vs	$V_s$	Algеб	$-V_s$
vref	$V_{ref}$	Algеб	$V_{ref0} - V_{ref}$
IN	$I_N$	Algеб	$u_e (-I_N y_{INT} + K_C X_{ad} I_{fd})$
FEX_	$y_{FEX}$	Algеб	$-y_{FEX} + \text{FixPiecewise} \left( (1, I_N \leq 0), (1 - 0.577 I_N, I_N \leq 0.433), \left( \sqrt{0.75 - I_N^2}, I_N \leq 0.75 \right) \right)$
vi	$V_i$	Algеб	$u_e (O_{EL} + U_{EL} - V_i + V_{ref} + V_s - y_{LG})$
LL_y	$y_{LL}$	Algеб	$T_B x'_{LL} - T_B y_{LL} + T_C (V_i - x'_{LL}) + \left( z_1^{LT_{LL}} \right)^2 (-x'_{LL} + y_{LL})$
HVG_	$y_{HVG}$	Algеб	$U_{EL} z_0^{None_{HVG}} - y_{HVG} + y_{LAZ} z_1^{None_{HVG}}$
LVG_	$y_{LVG}$	Algеб	$O_{EL} z_0^{None_{LVG}} + y_{HVG} z_1^{None_{LVG}} - y_{LVG}$
Se	$V_{out} * S_e( V_{out} )$	Algеб	$u_e \left( B_{SAT}^q z_0^{SL} \left( -A_{SAT}^q + y_{INT} \right)^2 - V_{out} * S_e( V_{out} ) \right)$
VFE	$V_{FE}$	Algеб	$u_e (K_D X_{ad} I_{fd} + K_E y_{INT} - V_{FE} + V_{out} * S_e( V_{out} ))$
WF_y	$y_{WF}$	Algеб	$K_F (V_{FE} - x'_{WF}) - T_{F1} y_{WF}$
vf	$v_f$	ExtAlgеб	$u_e (-v_{f0} + v_{out})$
Xad-	$X_{ad} I_{fd}$	ExtAlgеб	0
Ifd		ExtAlgеб	
a	$\theta$	ExtAlgеб	0
vbus	$V$	ExtAlgеб	0

## Services

Name	Symbol	Equation	Type
ue	$u_e$	$uu_g$	ConstService
UEL0	$UEL0$	-999	ConstService
OEL0	$OEL0$	999	ConstService
VA-MAXu	$VAMAXu$	$VAMAXu_e - 999u_e + 999$	ConstService
VAMINu	$VAMINu$	$VAMINu_e + 999u_e - 999$	ConstService
SAT_E1	$E_{SAT}^{1c}$	$E_1$	ConstService
SAT_E2	$E_{SAT}^{2c}$	$E_2 - 2z_{SAT}^{SE2} + 2$	ConstService
SAT_SE1	$SE_{SAT}^{1c}$	$S_{E1}$	ConstService
SAT_SE2	$SE_{SAT}^{2c}$	$S_{E2} - 2z_{SAT}^{SE2} + 2$	ConstService
SAT_a	$a_{SAT}$	$\sqrt{\frac{E_{SAT}^{1c} SE_{SAT}^{1c}}{E_{SAT}^{2c} SE_{SAT}^{2c}}} (\text{Indicator}(SE_{SAT}^{2c} > 0) + \text{Indicator}(SE_{SAT}^{2c} <$	ConstService
SAT_A	$A_{SAT}^q$	$E_{SAT}^{2c} - \frac{E_{SAT}^{1c} - E_{SAT}^{2c}}{a_{SAT}-1}$	ConstService
SAT_B	$B_{SAT}^q$	$\frac{E_{SAT}^{2c} SE_{SAT}^{2c} (a_{SAT}-1)^2 (\text{Indicator}(a_{SAT}>0)+\text{Indicator}(a_{SAT}<0))}{(E_{SAT}^{1c} - E_{SAT}^{2c})^2}$	ConstService
vref0	$V_{ref0}$	$V_{ref}$	PostInitService

## Discretes

Name	Symbol	Type	Info
LL_LT1	$LT_{LL}$	LessThan	
LL_LT2	$LT_{LL}$	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	<i>FEX</i>	Piecewise	Piecewise function FEX
LG	<i>LG</i>	Lag	Voltage transducer
LL	<i>LL</i>	LeadLag	V_A, Lead-lag compensator
LA	<i>LA</i>	LagAntiWindup	V_A, Anti-windup lag
HVG	<i>HVG</i>	HVGate	HVGate for under excitation
LVG	<i>LVG</i>	LVGate	HVGate for over excitation
SAT	<i>SAT</i>	ExcQuadSat	Field voltage saturation
INT	<i>INT</i>	Integrator	V_E, integrator
WF	<i>WF</i>	Washout	Stablizing circuit feedback

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)

### 5.12.16 ESST1A

Exciter ESST1A model.

Reference:

[1] PowerWorld, Exciter ESST1A, [Online],

[2] NEPLAN, Exciters Models, [Online],

Available: [https://www.powerworld.com/WebHelp/Content/TransientModels\\_HTML/Exciter%20ESST1A.htm](https://www.powerworld.com/WebHelp/Content/TransientModels_HTML/Exciter%20ESST1A.htm)

[https://www.neplan.ch/wp-content/uploads/2015/08/Nep\\_EXCITERS1.pdf](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		
VI-MAX	$V_{IMAX}$	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		
VA-MAX	$V_{AMAX}$	$V_A$ 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		
VR-MAX	$V_{RMAX}$	Maximum voltage regulator output limit	7.300	p.u.	
VR-MIN	$V_{RMIN}$	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 commuting 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		
ug	$u_g$	Generator online status	0	bool	
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	$y_{LG}$	State	State in lag transfer function	v_str	
LL_x	$x'_{LL}$	State	State in lead-lag	v_str	
LL1_x	$x'_{LL1}$	State	State in lead-lag	v_str	
LA_y	$y_{LA}$	State	State in lag TF	v_str	
WF_x	$x'_{WF}$	State	State in washout filter	v_str	
omega	$\omega$	ExtState	Generator speed		
v	$E_{term}$	Algeb	Input to exciter (bus v or Eterm)	v_str	
vout	$v_{out}$	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	$V_s$	Algeb	Voltage compensation from PSS	v_str	
vref	$V_{ref}$	Algeb	Reference voltage input	p.u.	v_str,v_iter
SG	$SG$	Algeb	SG	v_str	
LR_x	$x_{LR}$	Algeb	Value before limiter	v_str	
LR_y	$y_{LR}$	Algeb	Output after limiter and post gain	v_str	
vi	$V_i$	Algeb	Total input voltages	p.u.	v_str,v_iter
vil_x	$x_{vil}$	Algeb	Value before limiter	v_str	
vil_y	$y_{vil}$	Algeb	Output after limiter and post gain	v_str	
UEL2	$UEL_2$	Algeb	UEL_2 as HVG1 u1	v_str	
HVG1_y	$y_{HVG1}$	Algeb	HVGate output	v_str	
LL_y	$y_{LL}$	Algeb	Output of lead-lag	v_str	
LL1_y	$y_{LL1}$	Algeb	Output of lead-lag	v_str	
vas	$V_{As}$	Algeb	V_A after subtraction, as HVG u2	v_str,v_iter	
UEL3	$UEL_3$	Algeb	UEL_3 as HVG u1	v_str	
HVG_y	$y_{HVG}$	Algeb	HVGate output	v_str	
LVG_y	$y_{LVG}$	Algeb	LVGate output	v_str	
vol_x	$x_{vol}$	Algeb	Value before limiter	v_str	
vol_y	$y_{vol}$	Algeb	Output after limiter and post gain	v_str	
WF_y	$y_{WF}$	Algeb	Output of washout filter	v_str	
vf	$v_f$	ExtAlgeb	Excitation field voltage to generator		
XadIfd	$X_{ad}I_{fd}$	ExtAlgeb	Armature excitation current		
a	$\theta$	ExtAlgeb	Bus voltage phase angle		
vbus	$V$	ExtAlgeb	Bus voltage magnitude		
vd	$V_d$	ExtAlgeb	d-axis machine voltage		
vq	$V_q$	ExtAlgeb	q-axis machine voltage		

## Initialization Equations

Name	Symbol	Type	Initial Value
LG_y	$y_{LG}$	State	$E_{term}$
LL_x	$x'_{LL}$	State	$y_{HVG1}$
LL1_x	$x'_{LL1}$	State	$y_{LL}$
LA_y	$y_{LA}$	State	$K_A y_{LL1}$
WF_x	$x'_{WF}$	State	$y_{LVG}$
omega	$\omega$	ExtState	
v	$E_{term}$	Algeb	$V$
vout	$v_{out}$	Algeb	$u_e v_{f0}$
UEL	$U_{EL}$	Algeb	$UEL_0$
OEL	$O_{EL}$	Algeb	$OEL_0$
Vs	$V_s$	Algeb	0
vref	$V_{ref}$	Algeb	$u_e \left( E_{term} - SG s_1^{SWVOS} - U_{EL} s_1^{SWUEL} + \frac{-SG s_2^{SWVOS} + v_{f0} + y_{LR}}{K_A} \right)$
SG	$SG$	Algeb	$SG_0$
LR_x	$x_{LR}$	Algeb	$K_{LR} (-I_{LR} + X_{ad} I_{fd})$
LR_y	$y_{LR}$	Algeb	$x_{LR} z_i^{lim_{LR}} + z_l^{lim_{LR}} z_{zero}$
vi	$V_i$	Algeb	$u_e \left( SG s_1^{SWVOS} + U_{EL} s_1^{SWUEL} + V_{ref} + V_s - y_{LG} - y_{WF} \right)$
vil_x	$x_{vil}$	Algeb	$V_i$
vil_y	$y_{vil}$	Algeb	$V_{IMAX} z_u^{lim_{vil}} + V_{IMIN} z_l^{lim_{vil}} + x_{vil} z_i^{lim_{vil}}$
UEL2	$UEL_2$	Algeb	$u_e \left( U_{EL} s_2^{SWUEL} + llim \left( 1 - s_2^{SWUEL} \right) \right)$
HVG1_y	$y_{HVG1}$	Algeb	$UEL_2 z_0^{None_{HVG1}} + y_{vil} z_1^{None_{HVG1}}$
LL_y	$y_{LL}$	Algeb	$y_{HVG1}$
LL1_y	$y_{LL1}$	Algeb	$y_{LL}$
vas	$V_{As}$	Algeb	$u_e \left( SG s_2^{SWVOS} + y_{LA} - y_{LR} \right)$
UEL3	$UEL_3$	Algeb	$u_e \left( U_{EL} s_3^{SWUEL} + llim \left( 1 - s_3^{SWUEL} \right) \right)$
HVG_y	$y_{HVG}$	Algeb	$UEL_3 z_0^{None_{HVG}} + V_{As} z_1^{None_{HVG}}$
LVG_y	$y_{LVG}$	Algeb	$O_{EL} z_0^{None_{LVG}} + y_{HVG} z_1^{None_{LVG}}$
vol_x	$x_{vol}$	Algeb	$y_{LVG}$
vol_y	$y_{vol}$	Algeb	$efdl z_l^{lim_{vol}} + efd_u z_u^{lim_{vol}} + x_{vol} z_i^{lim_{vol}}$
WF_y	$y_{WF}$	Algeb	0
vf	$v_f$	ExtAlgeb	
XadIfd	$X_{ad} I_{fd}$	ExtAlgeb	
a	$\theta$	ExtAlgeb	
vbus	$V$	ExtAlgeb	
vd	$V_d$	ExtAlgeb	
vq	$V_q$	ExtAlgeb	

## Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
LG_y	$y_{LG}$	State	$E_{term} - y_{LG}$	$T_R$
LL_x	$x'_{LL}$	State	$-x'_{LL} + y_{HVG1}$	$T_B$
LL1_x	$x'_{LL1}$	State	$-x'_{LL1} + y_{LL}$	$T_{B1}$
LA_y	$y_{LA}$	State	$K_A y_{LL1} - y_{LA}$	$T_A$
WF_x	$x'_{WF}$	State	$-x'_{WF} + y_{LVG}$	$T_F$
omega	$\omega$	ExtState	0	

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
v	$E_{term}$	Algeb	$-E_{term} + V$
vout	$v_{out}$	Algeb	$uey_{vol} - v_{out}$
UEL	$UEL$	Algeb	$UEL_0 - U_{EL}$
OEL	$O_{EL}$	Algeb	$OEL_0 - O_{EL}$
Vs	$V_s$	Algeb	$-V_s$
vref	$V_{ref}$	Algeb	$V_{ref0} - V_{ref}$
SG	$SG$	Algeb	$-SG + SG_0$
LR_x	$x_{LR}$	Algeb	$K_{LR}(-I_{LR} + X_{ad}I_{fd}) - x_{LR}$
LR_y	$y_{LR}$	Algeb	$x_{LR}z_i^{lim_{LR}} - y_{LR} + z_l^{lim_{LR}} zero$
vi	$V_i$	Algeb	$-V_i + u_e \left( SGs_1^{SW_{VOS}} + U_{EL}s_1^{SW_{UEL}} + V_{ref} + V_s - y_{LG} - y_{WF} \right)$
vil_x	$x_{vil}$	Algeb	$V_i - x_{vil}$
vil_y	$y_{vil}$	Algeb	$V_{IMAX}z_u^{lim_{vil}} + V_{IMIN}z_l^{lim_{vil}} + x_{vil}z_i^{lim_{vil}} - y_{vil}$
UEL2	$UEL_2$	Algeb	$-UEL_2 + u_e \left( U_{EL}s_2^{SW_{UEL}} + llim \left( 1 - s_2^{SW_{UEL}} \right) \right)$
HVG1_y	$y_{HVG1}$	Algeb	$UEL_2z_0^{None_{HVG1}} - y_{HVG1} + y_{vil}z_1^{None_{HVG1}}$
LL_y	$y_{LL}$	Algeb	$T_Bx'_{LL} - T_By_{LL} + T_C(-x'_{LL} + y_{HVG1}) + \left( z_1^{LT_{LL}} \right)^2 (-x'_{LL} + y_{LL})$
LL1_y	$y_{LL1}$	Algeb	$T_B1x'_{LL1} - T_B1y_{LL1} + T_{C1}(-x'_{LL1} + y_{LL}) + \left( z_1^{LT_{LL1}} \right)^2 (-x'_{LL1} + y_{LL1})$
vas	$V_{As}$	Algeb	$-V_{As} + u_e \left( SGs_2^{SW_{VOS}} + y_{LA} - y_{LR} \right)$
UEL3	$UEL_3$	Algeb	$-UEL_3 + u_e \left( U_{EL}s_3^{SW_{UEL}} + llim \left( 1 - s_3^{SW_{UEL}} \right) \right)$
HVG_y	$y_{HVG}$	Algeb	$UEL_3z_0^{None_{HVG}} + V_{As}z_1^{None_{HVG}} - y_{HVG}$
LVG_y	$y_{LVG}$	Algeb	$O_{EL}z_0^{None_{LVG}} + y_{HVG}z_1^{None_{LVG}} - y_{LVG}$
vol_x	$x_{vol}$	Algeb	$-x_{vol} + y_{LVG}$
vol_y	$y_{vol}$	Algeb	$efdlz_l^{lim_{vol}} + efd_u z_u^{lim_{vol}} + x_{vol}z_i^{lim_{vol}} - y_{vol}$
WF_y	$y_{WF}$	Algeb	$K_F(-x'_{WF} + y_{LVG}) - T_Fy_{WF}$
vf	$v_f$	ExtAlgeb	$u_e(-v_{f0} + v_{out})$
XadIfd	$X_{ad}I_{fd}$	ExtAlgeb	0
a	$\theta$	ExtAlgeb	0
vbus	$V$	ExtAlgeb	0
vd	$V_d$	ExtAlgeb	0
vq	$V_q$	ExtAlgeb	0

## Services

Name	Symbol	Equation	Type
ue	$u_e$	$uu_g$	ConstService
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	$V_{A0}$	$-SGs_2^{SWVOS} + v_{f0} + y_{LR}$	PostInitService
vref0	$V_{ref0}$	$V_{ref}$	PostInitService
efdu	$efd_u$	$-K_C X_{ad} I_{fd} + V_{RMAX}  V_d + iV_q $	VarService
efdl	$efd_l$	$V_{RMIN}  V_d + iV_q $	VarService

## Discretes

Name	Symbol	Type	Info
SWUEL	$SWUEL$	Switcher	
SWVOS	$SWVOS$	Switcher	
LR_lim	$lim_{LR}$	HardLimiter	
vil_lim	$lim_{vil}$	HardLimiter	
HVG1_lt	$None_{HVG1}$	LessThan	
LL_LT1	$LT_{LL}$	LessThan	
LL_LT2	$LT_{LL}$	LessThan	
LL1_LT1	$LT_{LL1}$	LessThan	
LL1_LT2	$LT_{LL1}$	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	<i>LG</i>	Lag	Voltage transducer
LR	<i>LR</i>	GainLimiter	Exciter output current gain limiter
vil	<i>vil</i>	GainLimiter	Exciter voltage input limiter
HVG1	<i>HVG1</i>	HVGate	HVGate after V_I
LL	<i>LL</i>	LeadLag	Lead-lag compensator
LL1	<i>LL1</i>	LeadLag	Lead-lag compensator 1
LA	<i>LA</i>	LagAntiWindup	V_A, Anti-windup lag
HVG	<i>HVG</i>	HVGate	HVGate for under excitation
LVG	<i>LVG</i>	LVGate	HVGate for over excitation
vol	<i>vol</i>	GainLimiter	Exciter output limiter
WF	<i>WF</i>	Washout	V_F, Stablizing circuit feedback

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)

### 5.12.17 ESAC5A

Exciter ESAC5A.

## Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	$u$	connection status	1	<i>bool</i>	
name		device name			
syn		Synchronous generator idx			mandatory
TR	$T_R$	Sensing Time Constant	0.010	<i>p.u</i>	
TA	$T_A$	Voltage Regulator Time Constant	0.040	<i>p.u</i>	
KA	$K_A$	Voltage Regulator Gain	80		
VRMIN	$V_{Rmin}$	$V_R$ lower limit	-7.300	<i>p.u</i>	
VRMAX	$V_{Rmax}$	$V_R$ upper limit	7.300	<i>p.u</i>	
TE	$T_E$	Integrator Time Constant	0.800	<i>p.u</i>	non_negative
KF	$K_F$	Feedback Gain	0.030		
TF1	$T_{F1}$	Lag Time Constant	1	<i>p.u</i>	
TF2	$T_{F2}$	Lead-Lag Time Constant (pole)	0.800	<i>p.u</i>	
TF3	$T_{F3}$	Lead-Lag Time Constant (zero)	1	<i>p.u</i>	
KE	$K_E$	Exciter Feedback Gain	1		
E1	$E_1$	First saturation point	0	<i>p.u.</i>	
SE1	$S_E1$	Value at first saturation point	0	<i>p.u.</i>	
E2	$E_2$	Second saturation point	1	<i>p.u.</i>	
SE2	$S_E2$	Value at second saturation point	1	<i>p.u.</i>	
ug	$u_g$	Generator online status	0	<i>bool</i>	
Sn	$S_m$	Rated power from generator	0	<i>MVA</i>	
Vn	$V_m$	Rated voltage from generator	0	<i>kV</i>	
bus	$bus$	Bus idx of the generators	0		

## Variables

Name	Symbol	Type	Description	Unit	Properties
LP_y	$y_{LP}$	State	State in lag transfer function	v_str	
VR_y	$y_{VR}$	State	State in lag TF	v_str	
LL_x	$x'_{LL}$	State	State in lead-lag	v_str	
WF_x	$x'_{WF}$	State	State in washout filter	v_str	
INT_y	$y_{INT}$	State	Integrator output	v_str	
omega	$\omega$	ExtState	Generator speed		
v	$E_{term}$	Algeb	Input to exciter (bus v or Eterm)	v_str	
vout	$v_{out}$	Algeb	Exciter final output voltage	v_str	
UEL	$U_{EL}$	Algeb	Interface var for under exc. limiter	v_str	
OEL	$O_{EL}$	Algeb	Interface var for over exc. limiter	v_str	
Vs	$V_s$	Algeb	Voltage compensation from PSS	v_str	
vref	$V_{ref}$	Algeb	Reference voltage input	p.u.	v_str
vi	$vi$	Algeb	Total voltage input	pu	v_str
LL_y	$y_{LL}$	Algeb	Output of lead-lag	v_str	
WF_y	$y_{WF}$	Algeb	Output of washout filter	v_str	
Se	$V_{out} * S_e( V_{out} )$	Algeb	saturation output	v_str	
VFE	$V_{FE}$	Algeb	Combined saturation feedback	p.u.	v_str
vf	$v_f$	ExtAlgeb	Excitation field voltage to generator		
XadIfd	$X_{ad}I_{fd}$	ExtAlgeb	Armature excitation current		
a	$\theta$	ExtAlgeb	Bus voltage phase angle		
vbus	$V$	ExtAlgeb	Bus voltage magnitude		

## Initialization Equations

Name	Symbol	Type	Initial Value
LP_y	$y_{LP}$	State	$E_{term}$
VR_y	$y_{VR}$	State	$K_{Avi}$
LL_x	$x'_{LL}$	State	$y_{VR}$
WF_x	$x'_{WF}$	State	$y_{LL}$
INT_y	$y_{INT}$	State	$v_{f0}$
omega	$\omega$	ExtState	
v	$E_{term}$	Algeb	$V$
vout	$v_{out}$	Algeb	$u_e v_{f0}$
UEL	$U_{EL}$	Algeb	$UEL_0$
OEL	$O_{EL}$	Algeb	$OEL_0$
Vs	$V_s$	Algeb	0
vref	$V_{ref}$	Algeb	$E_{term} + \frac{V_{FE}}{K_A}$
vi	$vi$	Algeb	$u_e (-E_{term} + V_{ref})$
LL_y	$y_{LL}$	Algeb	$y_{VR}$
WF_y	$y_{WF}$	Algeb	0
Se	$V_{out} * S_e( V_{out} )$	Algeb	$B_{SAT}^q (-A_{SAT}^q + y_{INT})^2 \text{Indicator}(y_{INT} > A_{SAT}^q)$
VFE	$V_{FE}$	Algeb	$K_E y_{INT} + V_{out} * S_e( V_{out} )$
vf	$v_f$	ExtAlgeb	
XadIfd	$X_{ad} I_{fd}$	ExtAlgeb	
a	$\theta$	ExtAlgeb	
vbus	$V$	ExtAlgeb	

## Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
LP_y	$y_{LP}$	State	$E_{term} - y_{LP}$	$T_R$
VR_y	$y_{VR}$	State	$K_{Avi} - y_{VR}$	$T_A$
LL_x	$x'_{LL}$	State	$-x'_{LL} + y_{VR}$	$T_{F2}$
WF_x	$x'_{WF}$	State	$-x'_{WF} + y_{LL}$	$T_{F1}$
INT_y	$y_{INT}$	State	$u_e (-V_{FE} + y_{VR})$	$T_E$
omega	$\omega$	ExtState	0	

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
v	$E_{term}$	Algeb	$-E_{term} + V$
vout	$v_{out}$	Algeb	$ueyINT - v_{out}$
UEL	$UEL$	Algeb	$UEL_0 - UEL$
OEL	$OEL$	Algeb	$OEL_0 - OEL$
Vs	$V_s$	Algeb	$-V_s$
vref	$V_{ref}$	Algeb	$V_{ref0} - V_{ref}$
vi	$vi$	Algeb	$ue(V_{ref} + V_s - y_{LP} - y_{WF}) - vi$
LL_y	$y_{LL}$	Algeb	$T_{F2}x'_{LL} - T_{F2}y_{LL} + T_{F3}(-x'_{LL} + y_{VR}) + \left(z_1^{LT_{LL}}\right)^2(-x'_{LL} + y_{LL})$
WF_y	$y_{WF}$	Algeb	$K_F(-x'_{WF} + y_{LL}) - T_{F1}y_{WF}$
Se	$V_{out}$ $S_e( V_{out} )$	* Algeb	$ue\left(B_{SAT}^q z_0^{SL} (-A_{SAT}^q + y_{INT})^2 - V_{out} * S_e( V_{out} )\right)$
VFE	$V_{FE}$	Algeb	$ue(K_EyINT - V_{FE} + V_{out} * S_e( V_{out} ))$
vf	$v_f$	ExtAl- geb	$ue(-v_{f0} + v_{out})$
Xad- Ifd	$X_{ad}I_{fd}$	ExtAl- geb	0
a	$\theta$	ExtAl- geb	0
vbus	$V$	ExtAl- geb	0

## Services

Name	Symbol	Equation	Type
ue	$u_e$	$uu_g$	ConstService
UEL0	$UEL0$	0	ConstService
OEL0	$OEL0$	0	ConstService
VR-MAXu	$VRMAXu$	$V_{Rmax}u_e - 999u_e + 999$	ConstService
VRMINu	$VRMINu$	$V_{Rmin}u_e + 999u_e - 999$	ConstService
SAT_E1	$E_{SAT}^{1c}$	$E_1$	ConstService
SAT_E2	$E_{SAT}^{2c}$	$E_2 - 2z_{SAT}^{SE2} + 2$	ConstService
SAT_SE1	$SE_{SAT}^{1c}$	$S_{E1}$	ConstService
SAT_SE2	$SE_{SAT}^{2c}$	$S_{E2} - 2z_{SAT}^{SE2} + 2$	ConstService
SAT_a	$a_{SAT}$	$\sqrt{\frac{E_{SAT}^{1c}SE_{SAT}^{1c}}{E_{SAT}^{2c}SE_{SAT}^{2c}}} (\text{Indicator}(SE_{SAT}^{2c} > 0) + \text{Indicator}(SE_{SAT}^{2c} <$	ConstService
SAT_A	$A_{SAT}^q$	$E_{SAT}^{2c} - \frac{E_{SAT}^{1c} - E_{SAT}^{2c}}{a_{SAT}-1}$	ConstService
SAT_B	$B_{SAT}^q$	$\frac{E_{SAT}^{2c}SE_{SAT}^{2c}(a_{SAT}-1)^2(\text{Indicator}(a_{SAT}>0)+\text{Indicator}(a_{SAT}<0))}{(E_{SAT}^{1c} - E_{SAT}^{2c})^2}$	ConstService
vref0	$V_{ref0}$	$V_{ref}$	PostInitService

## Discretes

Name	Symbol	Type	Info
VR_lim	$lim_{VR}$	AntiWindup	Limiter in Lag
LL_LT1	$LT_{LL}$	LessThan	
LL_LT2	$LT_{LL}$	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

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)

## 5.13 Experimental

Experimental group

Common Parameters: u, name

## 5.14 FreqMeasurement

Frequency measurements.

Common Parameters: u, name

Common Variables: f

Available models: *BusFreq*, *BusROCOF*

### 5.14.1 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			
bus		bus idx			mandatory
Tf	$T_f$	input digital filter time const	0.020	sec	
Tw	$T_w$	washout time const	0.020	sec	
fn	$f_n$	nominal frequency	60	Hz	

## Variables

Name	Symbol	Type	Description	Unit	Properties
L_y	$y_L$	State	State in lag transfer function		v_str
WO_x	$x'_{WO}$	State	State in washout filter		v_str
WO_y	$y_{WO}$	Algeb	frequency deviation	p.u. (Hz)	v_str
f	$f$	Algeb	frequency output	p.u. (Hz)	v_str
a	$\theta$	ExtAlgeb			
v	$V$	ExtAlgeb			

## Initialization Equations

Name	Symbol	Type	Initial Value
L_y	$y_L$	State	$\theta - \theta_0$
WO_x	$x'_{WO}$	State	$y_L$
WO_y	$y_{WO}$	Algeb	0
f	$f$	Algeb	1
a	$\theta$	ExtAlgeb	
v	$V$	ExtAlgeb	

## Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
L_y	$y_L$	State	$\theta - \theta_0 - y_L$	$T_f$
WO_x	$x'_{WO}$	State	$-x'_{WO} + y_L$	$T_w$

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
WO_y	$y_{WO}$	Algeb	$1/\omega_n (-x'_{WO} + y_L) - T_w y_{WO}$
f	$f$	Algeb	$-f + y_{WO} + 1$
a	$\theta$	ExtAlgeb	0
v	$V$	ExtAlgeb	0

## Services

Name	Symbol	Equation	Type
iwn	$1/\omega_n$	$\frac{u}{2\pi f_n}$	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)

### 5.14.2 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			
bus		bus idx			mandatory
Tf	$T_f$	input digital filter time const	0.020	sec	
Tw	$T_w$	washout time const	0.020	sec	
fn	$f_n$	nominal frequency	60	Hz	
Tr	$T_r$	frequency washout time constant	0.100		

## Variables

Name	Symbol	Type	Description	Unit	Properties
L_y	$y_L$	State	State in lag transfer function		v_str
WO_x	$x'_{WO}$	State	State in washout filter		v_str
Wf_x	$x'_{Wf}$	State	State in washout filter		v_str
WO_y	$y_{WO}$	Algeb	frequency deviation	p.u. (Hz)	v_str
f	$f$	Algeb	frequency output	p.u. (Hz)	v_str
Wf_y	$y_{Wf}$	Algeb	Output of washout filter		v_str
a	$\theta$	ExtAlgeb			
v	$V$	ExtAlgeb			

## Initialization Equations

Name	Symbol	Type	Initial Value
L_y	$y_L$	State	$\theta - \theta_0$
WO_x	$x'_{WO}$	State	$y_L$
Wf_x	$x'_{Wf}$	State	$f$
WO_y	$y_{WO}$	Algeb	0
f	$f$	Algeb	1
Wf_y	$y_{Wf}$	Algeb	0
a	$\theta$	ExtAlgeb	
v	$V$	ExtAlgeb	

## Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
L_y	$y_L$	State	$\theta - \theta_0 - y_L$	$T_f$
WO_x	$x'_{WO}$	State	$-x'_{WO} + y_L$	$T_w$
Wf_x	$x'_{Wf}$	State	$f - x'_{Wf}$	$T_r$

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
WO_y	$y_{WO}$	Algeb	$1/\omega_n (-x'_{WO} + y_L) - T_w y_{WO}$
f	$f$	Algeb	$-f + y_{WO} + 1$
Wf_y	$y_{Wf}$	Algeb	$-T_r y_{Wf} + f - x'_{Wf}$
a	$\theta$	ExtAlgeb	0
v	$V$	ExtAlgeb	0

## Services

Name	Symbol	Equation	Type
iwn	$1/\omega_n$	$\frac{u}{2\pi f_n}$	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)

## 5.15 Information

Group for information container models.

Available models: [Summary](#)

### 5.15.1 Summary

Class for storing system summary. Can be used for random information or notes.

#### Parameters

Name	Symbol	Description	Default	Unit	Properties
field		field name			
comment		information, comment, or anything			
comment2		comment field 2			
comment3		comment field 3			
comment4		comment field 4			

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)

## 5.16 Interface

Group for interface models.

Available models: [Fortescue](#)

### 5.16.1 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- and 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	<i>bool</i>	
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	<i>MW</i>	non_zero
r	$r$	resistance	0.001	<i>p.u.</i>	<i>z</i>
x	$x$	short-circuit reactance	0.025	<i>p.u.</i>	non_zero,z
g		iron loss	0	<i>p.u.</i>	<i>y</i>
b		magnetizing susceptance	0.005	<i>p.u.</i>	<i>y</i>

## Variables

Name	Symbol	Type	Description	Unit	Properties
vp	$V_p$	Algeb	positive sequence voltage magnitude		v_str
ap	$\theta_p$	Algeb	positive sequence voltage phase		v_str
vnd	$V_{nd}$	Algeb	negative sequence voltage on d-axis (cos)		v_str
vnq	$V_{nq}$	Algeb	negative sequence voltage on q-axis (sin)		v_str
vzd	$V_{zd}$	Algeb	zero sequence voltage on d-axis (cos)		v_str
vzq	$V_{zq}$	Algeb	zero sequence voltage on q-axis (sin)		v_str
a	$\theta_1$	ExtAlgeb	phase angle of single-phase eq. bus		
aa	$\theta_a$	ExtAlgeb	phase angle of bus for phase a		
ab	$\theta_b$	ExtAlgeb	phase angle of bus for phase b		
ac	$\theta_c$	ExtAlgeb	phase angle of bus for phase c		
v	$V_1$	ExtAlgeb	voltage of single-phase eq. bus		
va	$V_a$	ExtAlgeb	voltage of bus for phase a		
vb	$V_b$	ExtAlgeb	voltage of bus for phase b		
vc	$V_c$	ExtAlgeb	voltage of bus for phase c		

## Initialization Equations

Name	Symbol	Type	Initial Value
vp	$V_p$	Algeb	$\frac{V_a}{3} + \frac{V_b}{3} + \frac{V_c}{3}$
ap	$\theta_p$	Algeb	$\theta_a + \theta_b + \theta_c$
vnd	$V_{nd}$	Algeb	0.0
vnq	$V_{nq}$	Algeb	0.0
vzd	$V_{zd}$	Algeb	0.0
vzq	$V_{zq}$	Algeb	0.0
a	$\theta_1$	ExtAlgeb	
aa	$\theta_a$	ExtAlgeb	
ab	$\theta_b$	ExtAlgeb	
ac	$\theta_c$	ExtAlgeb	
v	$V_1$	ExtAlgeb	
va	$V_a$	ExtAlgeb	
vb	$V_b$	ExtAlgeb	
vc	$V_c$	ExtAlgeb	

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
vp	$V_p$	Algeb	$-V_p + \frac{\sqrt{(V_b \sin(120^\circ - \theta_a + \theta_b) - V_c \sin(120^\circ + \theta_a - \theta_c))^2 + (V_a + V_b \cos(120^\circ - \theta_a + \theta_b) + V_c \cos(120^\circ + \theta_a - \theta_c))^2}}{3}$
ap	$\theta_p$	Algeb	$\theta_a - \theta_p + \text{atan2}(V_b \sin(120^\circ - \theta_a + \theta_b) - V_c \sin(120^\circ + \theta_a - \theta_c), V_a + V_b \cos(120^\circ - \theta_a + \theta_b) + V_c \cos(120^\circ + \theta_a - \theta_c))$
vnd	$V_{nd}$	Algeb	$V_a \cos(\theta_a) + V_b \cos(120^\circ - \theta_b) + V_c \cos(120^\circ + \theta_c) - V_{nd}$
vnq	$V_{nq}$	Algeb	$V_a \sin(\theta_a) - V_b \sin(120^\circ - \theta_b) + V_c \sin(120^\circ + \theta_c) - V_{nq}$
vzd	$V_{zd}$	Algeb	$V_a \cos(\theta_a) + V_b \cos(\theta_b) + V_c \cos(\theta_c) - V_{zd}$
vzq	$V_{zq}$	Algeb	$V_a \sin(\theta_a) + V_b \sin(\theta_b) + V_c \sin(\theta_c) - V_{zq}$
a	$\theta_1$	ExtAlgeb	$u(V_1^2(g + g_{hk}) - V_1 V_p(b_{hk} \sin(\theta_1 - \theta_p) + g_{hk} \cos(\theta_1 - \theta_p)))$
aa	$\theta_a$	ExtAlgeb	$\frac{u(-V_1 V_a(-b_{hk} \sin(\theta_1 - \theta_a) + g_{hk} \cos(\theta_1 - \theta_a)) + V_a^2(g + g_{hk}))}{3}$
ab	$\theta_b$	ExtAlgeb	$\frac{u(-V_1 V_b(b_{hk} \sin(120^\circ - \theta_1 + \theta_b) + g_{hk} \cos(120^\circ - \theta_1 + \theta_b)) + V_b^2(g + g_{hk}))}{3}$
ac	$\theta_c$	ExtAlgeb	$\frac{u(-V_1 V_c(-b_{hk} \sin(120^\circ + \theta_1 - \theta_c) + g_{hk} \cos(120^\circ + \theta_1 - \theta_c)) + V_c^2(g + g_{hk}))}{3}$
v	$V_1$	ExtAlgeb	$u(-V_1^2(b + b_{hk}) - V_1 V_p(-b_{hk} \cos(\theta_1 - \theta_p) + g_{hk} \sin(\theta_1 - \theta_p)))$
va	$V_a$	ExtAlgeb	$\frac{u(V_1 V_a(b_{hk} \cos(\theta_1 - \theta_a) + g_{hk} \sin(\theta_1 - \theta_a)) - V_a^2(b + b_{hk}))}{3}$
vb	$V_b$	ExtAlgeb	$\frac{u(V_1 V_b(b_{hk} \cos(120^\circ - \theta_1 + \theta_b) - g_{hk} \sin(120^\circ - \theta_1 + \theta_b)) - V_b^2(b + b_{hk}))}{3}$
vc	$V_c$	ExtAlgeb	$\frac{u(V_1 V_c(b_{hk} \cos(120^\circ + \theta_1 - \theta_c) + g_{hk} \sin(120^\circ + \theta_1 - \theta_c)) - V_c^2(b + b_{hk}))}{3}$

## Services

Name	Symbol	Equation	Type
yhk	$y_{hk}$	$\frac{u}{r+ix}$	ConstService
ghk	$g_{hk}$	$\text{re}(y_{hk})$	ConstService
bhk	$b_{hk}$	$\text{im}(y_{hk})$	ConstService
d120	$120^\circ$	$\frac{2\pi}{3}$	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)

## 5.17 Motor

Induction Motor group

Common Parameters: u, name

Available models: [Motor3](#), [Motor5](#)

### 5.17.1 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	<i>bool</i>	
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	<i>kWs/KVA</i>	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		

## Variables

Name	Symbol	Type	Description	Unit	Properties
slip	$\sigma$	State			v_str
e1d	$e'_d$	State	real part of 1st cage voltage		v_str
e1q	$e'_q$	State	imaginary part of 1st cage voltage		v_str
vd	$V_d$	Algeb	d-axis voltage		
vq	$V_q$	Algeb	q-axis voltage		
p	$P$	Algeb			v_str
q	$Q$	Algeb			v_str
Id	$I_d$	Algeb			v_str
Iq	$I_q$	Algeb			
te	$\tau_e$	Algeb			v_str
tm	$\tau_m$	Algeb			v_str
a	$\theta$	ExtAlgeb	Bus voltage phase angle		
v	$V$	ExtAlgeb	Bus voltage magnitude		

## Initialization Equations

Name	Symbol	Type	Initial Value
slip	$\sigma$	State	$1.0u$
e1d	$e'_d$	State	$0.05u$
e1q	$e'_q$	State	$0.9u$
vd	$V_d$	Algeb	
vq	$V_q$	Algeb	
p	$P$	Algeb	$u(I_dV_d + I_qV_q)$
q	$Q$	Algeb	$u(I_dV_q - I_qV_d)$
Id	$I_d$	Algeb	1
Iq	$I_q$	Algeb	
te	$\tau_e$	Algeb	$u(I_d e'_d + I_q e'_q)$
tm	$\tau_m$	Algeb	$u(\alpha + \beta\sigma + \sigma^2 c_2)$
a	$\theta$	ExtAlgeb	
v	$V$	ExtAlgeb	

## Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
slip	$\sigma$	State	$u(-\tau_e + \tau_m)$	$M$
e1d	$e'_d$	State	$u\left(\omega_b\sigma e'_q - \frac{I_q(-x' + x_0) + e'_d}{T'_0}\right)$	
e1q	$e'_q$	State	$u\left(-\omega_b\sigma e'_d - \frac{-I_d(-x' + x_0) + e'_q}{T'_0}\right)$	

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
vd	$V_d$	Algeb	$-Vu \sin(\theta) - V_d$
vq	$V_q$	Algeb	$Vu \cos(\theta) - V_q$
p	$P$	Algeb	$-P + u(I_dV_d + I_qV_q)$
q	$Q$	Algeb	$-Q + u(I_dV_q - I_qV_d)$
Id	$I_d$	Algeb	$u(-I_d r_s + I_q x' + V_d - e'_d)$
Iq	$I_q$	Algeb	$u(-I_d x' - I_q r_s + V_q - e'_q)$
te	$\tau_e$	Algeb	$-\tau_e + u(I_d e'_d + I_q e'_q)$
tm	$\tau_m$	Algeb	$-\tau_m + u(\alpha + \beta\sigma + \sigma^2 c_2)$
a	$\theta$	ExtAlgeb	$P$
v	$V$	ExtAlgeb	$Q$

## Services

Name	Symbol	Equation	Type
wb	$\omega_b$	$2\pi f$	ConstService
x0	$x_0$	$x_m + x_s$	ConstService
x1	$x'$	$\frac{x_m x_{R1}}{x_m + x_{R1}} + x_s$	ConstService
T10	$T'_0$	$\frac{x_m + x_{R1}}{\omega_b r_{R1}}$	ConstService
M	$M$	$2H_m$	ConstService
aa	$\alpha$	$c_1 + c_2 + c_3$	ConstService
bb	$\beta$	$-c_2 - 2c_3$	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)

## 5.17.2 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	<i>bool</i>	
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	<i>kWs/KVA</i>	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		

## Variables

Name	Symbol	Type	Description	Unit	Properties
slip	$\sigma$	State			v_str
e1d	$e'_d$	State	real part of 1st cage voltage		v_str
e1q	$e'_q$	State	imaginary part of 1st cage voltage		v_str
e2d	$e''_d$	State	real part of 2nd cage voltage		v_str
e2q	$e''_q$	State	imag part of 2nd cage voltage		v_str
vd	$V_d$	Algeb	d-axis voltage		
vq	$V_q$	Algeb	q-axis voltage		
p	$P$	Algeb			v_str
q	$Q$	Algeb			v_str
Id	$I_d$	Algeb			v_str
Iq	$I_q$	Algeb			v_str
te	$\tau_e$	Algeb			v_str
tm	$\tau_m$	Algeb			v_str
a	$\theta$	ExtAlgeb	Bus voltage phase angle		
v	$V$	ExtAlgeb	Bus voltage magnitude		

## Initialization Equations

Name	Symbol	Type	Initial Value
slip	$\sigma$	State	$1.0u$
e1d	$e'_d$	State	$0.05u$
e1q	$e'_q$	State	$0.9u$
e2d	$e''_d$	State	$0.05u$
e2q	$e''_q$	State	$0.9u$
vd	$V_d$	Algeb	
vq	$V_q$	Algeb	
p	$P$	Algeb	$u(I_dV_d + I_qV_q)$
q	$Q$	Algeb	$u(I_dV_q - I_qV_d)$
Id	$I_d$	Algeb	$0.9u$
Iq	$I_q$	Algeb	$0.1u$
te	$\tau_e$	Algeb	$u(I_d e''_d + I_q e''_q)$
tm	$\tau_m$	Algeb	$u(\alpha + \beta\sigma + \sigma^2 c_2)$
a	$\theta$	ExtAlgeb	
v	$V$	ExtAlgeb	

## Differential Equations

Name	Sym- bol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
slip	$\sigma$	State	$u(-\tau_e + \tau_m)$	$M$
e1d	$e'_d$	State	$u\left(\omega_b\sigma e'_q - \frac{I_q(-x'+x_0)+e'_d}{T'_0}\right)$	
e1q	$e'_q$	State	$u\left(-\omega_b\sigma e'_d - \frac{-I_d(-x'+x_0)+e'_q}{T'_0}\right)$	
e2d	$e''_d$	State	$u\left(\omega_b\sigma e'_q - \omega_b\sigma(-e''_q + e'_q) - \frac{I_q(-x'+x_0)+e'_d}{T'_0} + \frac{-I_q(x'-x'')-e''_d+e'_d}{T''_0}\right)$	
e2q	$e''_q$	State	$u\left(-\omega_b\sigma e'_d + \omega_b\sigma(-e''_d + e'_d) - \frac{-I_d(-x'+x_0)+e'_q}{T'_0} + \frac{I_d(x'-x'')-e''_q+e'_q}{T''_0}\right)$	

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
vd	$V_d$	Algeb	$-Vu \sin(\theta) - V_d$
vq	$V_q$	Algeb	$Vu \cos(\theta) - V_q$
p	$P$	Algeb	$-P + u(I_d V_d + I_q V_q)$
q	$Q$	Algeb	$-Q + u(I_d V_q - I_q V_d)$
Id	$I_d$	Algeb	$u(-I_d r_s + I_q x'' + V_d - e_d'')$
Iq	$I_q$	Algeb	$u(-I_d x'' - I_q r_s + V_q - e_q'')$
te	$\tau_e$	Algeb	$-\tau_e + u(I_d e_d'' + I_q e_q'')$
tm	$\tau_m$	Algeb	$-\tau_m + u(\alpha + \beta\sigma + \sigma^2 c_2)$
a	$\theta$	ExtAlgeb	$P$
v	$V$	ExtAlgeb	$Q$

## Services

Name	Symbol	Equation	Type
wb	$\omega_b$	$2\pi f$	ConstService
x0	$x_0$	$x_m + x_s$	ConstService
x1	$x'$	$\frac{x_m x_{R1}}{x_m + x_{R1}} + x_s$	ConstService
T10	$T'_0$	$\frac{x_m}{\omega_b r_{R1}}$	ConstService
M	$M$	$2H_m$	ConstService
aa	$\alpha$	$c_1 + c_2 + c_3$	ConstService
bb	$\beta$	$-c_2 - 2c_3$	ConstService
x2	$x''$	$\frac{x_m x_{R1} x_{R2}}{x_m x_{R1} + x_m x_{R2} + x_{R1} x_{R2}} + x_s$	ConstService
T20	$T''_0$	$\frac{x_m}{\omega_b r_{R2}}$	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)

## 5.18 OutputSelect

Group for selecting outputs.

Available models: *Output*

### 5.18.1 Output

Model for specifying output models and/or variables.

#### Parameters

Name	Symbol	Description	Default	Unit	Properties
model		Name of the model			mandatory
varname		Variable name			
dev		Device name			

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)

## 5.19 PLL

Phase-locked loop models.

Common Parameters: u, name

Common Variables: am

Available models: *PLL1*, *PLL2*

### 5.19.1 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 ( $K_p, K_i$ ) -> 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	

#### Variables

Name	Symbol	Type	Description	Unit	Properties
af_y	$y_{af}$	State	State in lag transfer function	v_str	
PI_xi	$xi_{PI}$	State	Integrator output	v_str	
ae	$\theta_{est}$	State	PLL angle output before filter	v_str	
am	$\theta_m$	State	PLL output angle after filtering	v_str	
PI_y	$y_{PI}$	Algeb	PI output	v_str	
a	$\theta$	ExtAlgeb	Bus voltage angle		

## Initialization Equations

Name	Symbol	Type	Initial Value
af_y	$y_{af}$	State	$\theta$
PI_xi	$xi_{PI}$	State	0.0
ae	$\theta_{est}$	State	$\theta$
am	$\theta_m$	State	$\theta$
PI_y	$y_{PI}$	Algeb	$K_p u (-\theta_m + y_{af})$
a	$\theta$	ExtAlgeb	

## Differential Equations

Name	Symbol	Type	RHS of Equation " $T \dot{x} = f(x, y)$ "	T (LHS)
af_y	$y_{af}$	State	$\theta - y_{af}$	$T_f$
PI_xi	$xi_{PI}$	State	$K_i u (-\theta_m + y_{af})$	
ae	$\theta_{est}$	State	$2\pi f_n y_{PI}$	
am	$\theta_m$	State	$\theta_{est} - \theta_m$	$T_p$

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
PI_y	$y_{PI}$	Algeb	$K_p u (-\theta_m + y_{af}) + xi_{PI} - y_{PI}$
a	$\theta$	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)

## 5.19.2 PLL2

Synchronously-rotating Reference Frame (SRF) Phasor Lock Loop (PLL).

The PLL minimizes  $vq = v \sin(\alpha - \alpha_m)$  using a PI controller.

The output signal is  $\alpha_m$ , a state variable.

### Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	$u$	connection status	1	<i>bool</i>	
name		device name			
bus		bus idx			mandatory
fn	$f_n$	nominal frequency	60	<i>Hz</i>	
Kp	$K_p$	proportional gain	0.100		
Ki	$K_i$	integral gain	0.100		

### Variables

Name	Symbol	Type	Description	Unit	Properties
PI_xi	$xi_{PI}$	State	Integrator output		v_str
am	$\theta_m$	State	PLL angle output		v_str
PI_y	$y_{PI}$	Algeb	PI output		v_str
a	$\theta$	ExtAlgeb	Bus voltage angle		
v	$V$	ExtAlgeb	Bus voltage magnitude		

### Initialization Equations

Name	Symbol	Type	Initial Value
PI_xi	$xi_{PI}$	State	0
am	$\theta_m$	State	$\theta$
PI_y	$y_{PI}$	Algeb	$K_p V \sin(\theta - \theta_m)$
a	$\theta$	ExtAlgeb	
v	$V$	ExtAlgeb	

## Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
PI_xi	$xi_{PI}$	State	$K_i V \sin(\theta - \theta_m)$	
am	$\theta_m$	State	$2\pi f_n y_{PI}$	

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
PI_y	$y_{PI}$	Algeb	$K_p V \sin(\theta - \theta_m) + xi_{PI} - y_{PI}$
a	$\theta$	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)

## 5.20 PSS

Power system stabilizer group.

Common Parameters: u, name

Common Variables: vsout

Available models: [IEEEEST](#), [ST2CUT](#)

### 5.20.1 IEEEEST

IEEEEST 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	<i>bool</i>	
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	$L_{S MAX}$	Max. output limit	0.300		
LSMIN	$L_{S MIN}$	Min. output limit	-0.300		
VCU	$V_{CU}$	Upper enabling bus voltage	999	<i>p.u.</i>	
VCL	$V_{CL}$	Upper enabling bus voltage	-999	<i>p.u.</i>	
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	$x'_{F1}$	State	State in 2nd order LPF	v_str	
F1_y	$y_{F1}$	State	Output of 2nd order LPF	v_str	
F2_x1	$x'_{F2}$	State	State #1 in 2nd order lead-lag	v_str	
F2_x2	$x''_{F2}$	State	State #2 in 2nd order lead-lag	v_str	
LL1_x	$x'_{LL1}$	State	State in lead-lag	v_str	
LL2_x	$x'_{LL2}$	State	State in lead-lag	v_str	
WO_x	$x'_{WO}$	State	State in washout filter	v_str	
omega	$\omega$	ExtState	Generator speed	p.u.	
vsout	$v_{sout}$	Algeb	PSS output voltage to exciter		
sig	$S_g$	Algeb	Input signal	v_str	
F2_y	$y_{F2}$	Algeb	Output of 2nd order lead-lag	v_str	
LL1_y	$y_{LL1}$	Algeb	Output of lead-lag	v_str	
LL2_y	$y_{LL2}$	Algeb	Output of lead-lag	v_str	
Vks_y	$y_{Vks}$	Algeb	Gain output	v_str	
WO_y	$y_{WO}$	Algeb	Output of washout filter	v_str	
Vss	$V_{ss}$	Algeb	Voltage output before output limiter		
tm	$\tau_m$	ExtAlgeb	Generator mechanical input		
te	$\tau_e$	ExtAlgeb	Generator electrical output		
v	$V$	ExtAlgeb	Bus (or busr, if given) terminal voltage		
f	$f$	ExtAlgeb	Bus frequency		
vi	$v_i$	ExtAlgeb	Exciter input voltage		

## Initialization Equations

Name	Symbol	Type	Initial Value
F1_x	$x'_{F1}$	State	0
F1_y	$y_{F1}$	State	$S_{ig}$
F2_x1	$x'_{F2}$	State	0
F2_x2	$x''_{F2}$	State	$y_{F1}$
LL1_x	$x'_{LL1}$	State	$y_{F2}$
LL2_x	$x'_{LL2}$	State	$y_{LL1}$
WO_x	$x'_{WO}$	State	$y_{Vks}$
omega	$\omega$	ExtState	
vsout	$v_{sout}$	Algeb	
sig	$S_{ig}$	Algeb	$u_e \left( V s_5^{SW} + s_1^{SW} (\omega - 1) + s_4^{SW} (\tau_m - \tau_{m0}) + \frac{\tau_{m0} s_3^{SW}}{(Sb/Sn)} \right)$
F2_y	$y_{F2}$	Algeb	$y_{F1}$
LL1_y	$y_{LL1}$	Algeb	$y_{F2}$
LL2_y	$y_{LL2}$	Algeb	$y_{LL1}$
Vks_y	$y_{Vks}$	Algeb	$K_{SYLL2}$
WO_y	$y_{WO}$	Algeb	$x'_{WO} z_1^{LT_{WO}}$
Vss	$V_{ss}$	Algeb	
tm	$\tau_m$	ExtAlgeb	
te	$\tau_e$	ExtAlgeb	
v	$V$	ExtAlgeb	
f	$f$	ExtAlgeb	
vi	$v_i$	ExtAlgeb	

## Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
F1_x	$x'_{F1}$	State	$-A_1 x'_{F1} + S_{ig} - y_{F1}$	$A_2$
F1_y	$y_{F1}$	State	$x'_{F1}$	
F2_x1	$x'_{F2}$	State	$-A_3 x'_{F2} - x''_{F2} + y_{F1}$	$A_4$
F2_x2	$x''_{F2}$	State	$x'_{F2}$	
LL1_x	$x'_{LL1}$	State	$-x'_{LL1} + y_{F2}$	$T_2$
LL2_x	$x'_{LL2}$	State	$-x'_{LL2} + y_{LL1}$	$T_4$
WO_x	$x'_{WO}$	State	$-x'_{WO} + y_{Vks}$	$T_6$
omega	$\omega$	ExtState	0	

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
vsout	$v_{sout}$	Algeb	$V_{ss}u_e z_i^{OLIM} - v_{sout}$
sig	$S_{ig}$	Algeb	$-S_{ig} + u_e \left( V s_5^{SW} + s_1^{SW} (\omega - 1) + s_2^{SW} (f - 1) + s_4^{SW} (\tau_m - \tau_{m0}) + s_6^{SW} v^{dV/dt} + \frac{\tau_e s_3^{SW}}{(Sb/Sn)} \right)$
F2_y	$y_{F2}$	Algeb	$A_4 A_5 x'_{F2} + A_4 x''_{F2} - A_4 y_{F2} + A_6 (-A_3 x'_{F2} - x''_{F2} + y_{F1}) + \left( z_1^{LT_{F2}} \right)^4 (-x''_{F2} + y_{F2})$
LL1_y	$y_{LL1}$	Algeb	$T_1 (-x'_{LL1} + y_{F2}) + T_2 x'_{LL1} - T_2 y_{LL1} + \left( z_1^{LT_{LL1}} \right)^2 (-x'_{LL1} + y_{LL1})$
LL2_y	$y_{LL2}$	Algeb	$T_3 (-x'_{LL2} + y_{LL1}) + T_4 x'_{LL2} - T_4 y_{LL2} + \left( z_1^{LT_{LL2}} \right)^2 (-x'_{LL2} + y_{LL2})$
Vks_y	$y_{Vks}$	Algeb	$K_{SYLL2} - y_{Vks}$
WO_y	$y_{WO}$	Algeb	$T_5 z_0^{LT_{WO}} (-x'_{WO} + y_{Vks}) + T_6 x'_{WO} z_1^{LT_{WO}} - T_6 y_{WO}$
Vss	$V_{ss}$	Algeb	$L_{SMAX} z_u^{VLIM} + L_{SMIN} z_l^{VLIM} - V_{ss} + y_{WO} z_i^{VLIM}$
tm	$\tau_m$	ExtAlgeb	0
te	$\tau_e$	ExtAlgeb	0
v	$V$	ExtAlgeb	0
f	$f$	ExtAlgeb	0
vi	$v_i$	ExtAlgeb	$u_e v_{sout}$

## Services

Name	Symbol	Equation	Type
ue	$u_e$	$uu_{ee}$	ConstService

## Discretes

Name	Symbol	Type	Info
dv	$dV/dt$	Derivative	Finite difference of bus voltage
SW	$SW$	Switcher	
F2_LT1	$LT_{F2}$	LessThan	
F2_LT2	$LT_{F2}$	LessThan	
F2_LT3	$LT_{F2}$	LessThan	
F2_LT4	$LT_{F2}$	LessThan	
LL1_LT1	$LT_{LL1}$	LessThan	
LL1_LT2	$LT_{LL1}$	LessThan	
LL2_LT1	$LT_{LL2}$	LessThan	
LL2_LT2	$LT_{LL2}$	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 [IEEEEST]

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',)

## 5.20.2 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	<i>u</i>	connection status	1	<i>bool</i>	
name		device name			
avr		Exciter idx			mandatory
MODE		Input signal 1			mandatory
busr		Remote bus 1			
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	<i>K</i> <sub>1</sub>	Transducer 1 gain	1		
K2	<i>K</i> <sub>2</sub>	Transducer 2 gain	1		
T1	<i>T</i> <sub>1</sub>	Transducer 1 time const.	1		
T2	<i>T</i> <sub>2</sub>	Transducer 2 time const.	1		
T3	<i>T</i> <sub>3</sub>	Washout int. time const.	1		
T4	<i>T</i> <sub>4</sub>	Washout delay time const.	0.200		
T5	<i>T</i> <sub>5</sub>	Leadlag 1 time const. (1)	1		
T6	<i>T</i> <sub>6</sub>	Leadlag 1 time const. (2)	0.500		
T7	<i>T</i> <sub>7</sub>	Leadlag 2 time const. (1)	1		
T8	<i>T</i> <sub>8</sub>	Leadlag 2 time const. (2)	1		
T9	<i>T</i> <sub>9</sub>	Leadlag 3 time const. (1)	1		
T10	<i>T</i> <sub>10</sub>	Leadlag 3 time const. (2)	0.200		
LSMAX	<i>L</i> <sub>SMAX</sub>	Max. output limit	0.300		
LSMIN	<i>L</i> <sub>SMIN</sub>	Min. output limit	-0.300		
VCU	<i>V</i> <sub>CU</sub>	Upper enabling bus voltage	999	p.u.	
VCL	<i>V</i> <sub>CL</sub>	Upper enabling bus voltage	-999	p.u.	
syn		Retrieved generator idx	0		
bus		Retrieved bus idx			
Sn	<i>S</i> <sub>n</sub>	Generator power base	0		

## Variables

Name	Symbol	Type	Description	Unit	Properties
L1_y	$y_{L1}$	State	State in lag transfer function		v_str
L2_y	$y_{L2}$	State	State in lag transfer function		v_str
WO_x	$x'_{WO}$	State	State in washout filter		v_str
LL1_x	$x'_{LL1}$	State	State in lead-lag		v_str
LL2_x	$x'_{LL2}$	State	State in lead-lag		v_str
LL3_x	$x'_{LL3}$	State	State in lead-lag		v_str
omega	$\omega$	ExtState	Generator speed	p.u.	
vsout	$v_{sout}$	Algeb	PSS output voltage to exciter		
sig	$S_{ig}$	Algeb	Input signal		v_str
sig2	$S_{ig2}$	Algeb	Input signal 2		v_str
IN	$I_N$	Algeb	Sum of inputs		v_str
WO_y	$y_{WO}$	Algeb	Output of washout filter		v_str
LL1_y	$y_{LL1}$	Algeb	Output of lead-lag		v_str
LL2_y	$y_{LL2}$	Algeb	Output of lead-lag		v_str
LL3_y	$y_{LL3}$	Algeb	Output of lead-lag		v_str
VSS_X	$x_{VSS}$	Algeb	Value before limiter		v_str
VSS_y	$y_{VSS}$	Algeb	Output after limiter and post gain		v_str
tm	$\tau_m$	ExtAlgeb	Generator mechanical input		
te	$\tau_e$	ExtAlgeb	Generator electrical output		
v	$V$	ExtAlgeb	Bus (or busr, if given) terminal voltage		
f	$f$	ExtAlgeb	Bus frequency		
vi	$v_i$	ExtAlgeb	Exciter input voltage		
v2	$V$	ExtAlgeb	Bus (or busr2, if given) terminal voltage		
f2	$f_2$	ExtAlgeb	Bus frequency 2		

## Initialization Equations

Name	Symbol	Type	Initial Value
L1_y	$y_{L1}$	State	$K_1 S_{ig}$
L2_y	$y_{L2}$	State	$K_2 S_{ig2}$
WO_x	$x'_{WO}$	State	$I_N$
LL1_x	$x'_{LL1}$	State	$y_{WO}$
LL2_x	$x'_{LL2}$	State	$y_{LL1}$
LL3_x	$x'_{LL3}$	State	$y_{LL2}$
omega	$\omega$	ExtState	
vsout	$v_{sout}$	Algeb	
sig	$S_{ig}$	Algeb	$V s_5^{SW} + s_1^{SW} (\omega - 1) + s_4^{SW} (\tau_m - \tau_{m0}) + \frac{\tau_{m0} s_3^{SW}}{(Sb/Sn)}$
sig2	$S_{ig2}$	Algeb	$V s_5^{SW2} + s_1^{SW2} (\omega - 1) + s_4^{SW2} (\tau_m - \tau_{m0}) + \frac{\tau_{m0} s_3^{SW2}}{(Sb/Sn)}$
IN	$I_N$	Algeb	$u_e (y_{L1} + y_{L2})$
WO_y	$y_{WO}$	Algeb	$x'_{WO} z_1^{LTWO}$
LL1_y	$y_{LL1}$	Algeb	$y_{WO}$
LL2_y	$y_{LL2}$	Algeb	$y_{LL1}$
LL3_y	$y_{LL3}$	Algeb	$y_{LL2}$
VSS_x	$x_{VSS}$	Algeb	$y_{LL3}$
VSS_y	$y_{VSS}$	Algeb	$L_{SMAX} z_u^{lim_{VSS}} + L_{SMIN} z_l^{lim_{VSS}} + x_{VSS} z_i^{lim_{VSS}}$
tm	$\tau_m$	ExtAlgeb	
te	$\tau_e$	ExtAlgeb	
v	$V$	ExtAlgeb	
f	$f$	ExtAlgeb	
vi	$v_i$	ExtAlgeb	
v2	$V$	ExtAlgeb	
f2	$f_2$	ExtAlgeb	

## Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
L1_y	$y_{L1}$	State	$K_1 S_{ig} - y_{L1}$	$T_1$
L2_y	$y_{L2}$	State	$K_2 S_{ig2} - y_{L2}$	$T_2$
WO_x	$x'_{WO}$	State	$I_N - x'_{WO}$	$T_4$
LL1_x	$x'_{LL1}$	State	$-x'_{LL1} + y_{WO}$	$T_6$
LL2_x	$x'_{LL2}$	State	$-x'_{LL2} + y_{LL1}$	$T_8$
LL3_x	$x'_{LL3}$	State	$-x'_{LL3} + y_{LL2}$	$T_{10}$
omega	$\omega$	ExtState	0	

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
vsout	$v_{sout}$	Algeb	$u_e y_{VSS} z_i^{OLIM} - v_{sout}$
sig	$S_{ig}$	Algeb	$-S_{ig} + V s_5^{SW} + s_1^{SW} (\omega - 1) + s_2^{SW} (f - 1) + s_4^{SW} (\tau_m - \tau_{m0}) + s_6^{SW} v^{dv} + \frac{\tau_e s_3^{SW}}{(Sb/Sn)}$
sig2	$S_{ig2}$	Algeb	$-S_{ig2} + V s_5^{SW2} + s_1^{SW2} (\omega - 1) + s_2^{SW2} (f_2 - 1) + s_4^{SW2} (\tau_m - \tau_{m0}) + s_6^{SW2} v^{dv2} + \frac{\tau_e s_3^{SW2}}{(Sb/Sn)}$
IN	$I_N$	Algeb	$-I_N + u_e (y_{L1} + y_{L2})$
WO_y	$y_{WO}$	Algeb	$T_3 z_0^{LT_{WO}} (I_N - x'_{WO}) + T_4 x'_{WO} z_1^{LT_{WO}} - T_4 y_{WO}$
LL1_y	$y_{LL1}$	Algeb	$T_5 (-x'_{LL1} + y_{WO}) + T_6 x'_{LL1} - T_6 y_{LL1} + (z_1^{LT_{LL1}})^2 (-x'_{LL1} + y_{LL1})$
LL2_y	$y_{LL2}$	Algeb	$T_7 (-x'_{LL2} + y_{LL1}) + T_8 x'_{LL2} - T_8 y_{LL2} + (z_1^{LT_{LL2}})^2 (-x'_{LL2} + y_{LL2})$
LL3_y	$y_{LL3}$	Algeb	$T_9 (-x'_{LL3} + y_{LL2}) + T_{10} x'_{LL3} - T_{10} y_{LL3} + (z_1^{LT_{LL3}})^2 (-x'_{LL3} + y_{LL3})$
VSS_x	$x_{VSS}$	Algeb	$-x_{VSS} + y_{LL3}$
VSS_y	$y_{VSS}$	Algeb	$L_{SMAX} z_u^{lim_{VSS}} + L_{SMIN} z_l^{lim_{VSS}} + x_{VSS} z_i^{lim_{VSS}} - y_{VSS}$
tm	$\tau_m$	ExtAlgeb	0
te	$\tau_e$	ExtAlgeb	0
v	$V$	ExtAlgeb	0
f	$f$	ExtAlgeb	0
vi	$v_i$	ExtAlgeb	$u_e v_{sout}$
v2	$V$	ExtAlgeb	0
f2	$f_2$	ExtAlgeb	0

## Services

Name	Symbol	Equation	Type
ue	$u_e$	$uu_{ee}$	ConstService
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_LT1	$LT_{LL1}$	LessThan	
LL1_LT2	$LT_{LL1}$	LessThan	
LL2_LT1	$LT_{LL2}$	LessThan	
LL2_LT2	$LT_{LL2}$	LessThan	
LL3_LT1	$LT_{LL3}$	LessThan	
LL3_LT2	$LT_{LL3}$	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',)

## 5.21 PhasorMeasurement

Phasor measurements

Common Parameters: u, name

Common Variables: am, vm

Available models: *PMU*

### 5.21.1 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	<i>bool</i>	
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		

#### Variables

Name	Symbol	Type	Description	Unit	Properties
am	$\theta_m$	State	phase angle measurement	<i>rad.</i>	v_str
vm	$V_m$	State	voltage magnitude measurement	<i>p.u.(kV)</i>	v_str
a	$\theta$	ExtAlgeb	Bus voltage phase angle		
v	$V$	ExtAlgeb	Bus voltage magnitude		

#### Initialization Equations

Name	Symbol	Type	Initial Value
am	$\theta_m$	State	$\theta$
vm	$V_m$	State	$V$
a	$\theta$	ExtAlgeb	
v	$V$	ExtAlgeb	

## Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
am	$\theta_m$	State	$\theta - \theta_m$	$T_a$
vm	$V_m$	State	$V - V_m$	$T_v$

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
a	$\theta$	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)

## 5.22 RenAerodynamics

Renewable aerodynamics group.

Common Parameters: u, name, rego

Common Variables: theta

Available models: [WTARA1](#), [WTARV1](#)

### 5.22.1 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	$\theta_0$	Initial pitch angle	0	deg.	

## Variables

Name	Symbol	Type	Description	Unit	Properties
theta	$\theta$	Algeb	Pitch angle	rad	v_str
Pmg	$P_{mg}$	ExtAlgeb			

## Initialization Equations

Name	Symbol	Type	Initial Value
theta	$\theta$	Algeb	$\theta_{0r}$
Pmg	$P_{mg}$	ExtAlgeb	

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
theta	$\theta$	Algeb	$-\theta + \theta_{0r}$
Pmg	$P_{mg}$	ExtAlgeb	$-K_a\theta(\theta - \theta_0)$

## Services

Name	Symbol	Equation	Type
theta0r	$\theta_{0r}$	$\frac{\pi\theta_0}{180}$	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)

## 5.22.2 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	<i>bool</i>	
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	<i>m</i>	
ngb		gear box ratio	5		
rho		air density	1.200	<i>kg/m3</i>	
Sn	$S_n$		0		

### Variables

Name	Symbol	Type	Description	Unit	Properties
theta	$\theta$	Algeb	Pitch angle	<i>rad</i>	
Pmg	$P_{mg}$	ExtAlgeb			

## Initialization Equations

Name	Symbol	Type	Initial Value
theta	$\theta$	Algeb	
Pmg	$P_{mg}$	ExtAlgeb	

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
theta	$\theta$	Algeb	0
Pmg	$P_{mg}$	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)

## 5.23 RenExciter

Renewable electrical control (exciter) group.

Common Parameters: u, name, reg

Common Variables: Pref, Qref, wg, Pord

Available models: [REECA1](#), [REECA1E](#), [REECA1G](#)

### 5.23.1 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.

- $K_{qv}$  in pu current / pu voltage deviation controls the intensity of reactive power injection. The parameter needs to be tuned properly to avoid voltage overshoot.
- When multiple renewable generators are connected to the same bus,  $K_{qv}$  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	
QMax	$Q_{max}$	Upper limit for reactive power regulator	999		
QMin	$Q_{min}$	Lower limit for reactive power regulator	-999		
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	$d_{P_{max}}$	Power reference max. ramp rate ( $>0$ )	999		
dPmin	$d_{P_{in}}$	Power reference min. ramp rate ( $<0$ )	-999		
PMAX	$P_{max}$	Max. active power limit $> 0$	999		

continues on next page

Table 17 – continued from previous page

Name	Symbol	Description	Default	Unit	Properties
PMIN	$P_{min}$	Min. active power limit	0		
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
bus		Retrieved bus idx			
gen		Retrieved StaticGen idx			
Sn	$S_n$		0		

## Variables

Name	Symbol	Type	Description	Unit	Properties
s0_y	$y_{s0}$	State	State in lag transfer function	v_str	
S1_y	$y_{S_1}$	State	State in lag transfer function	v_str	
PIQ_xi	$xi_{PIQ}$	State	Integrator output	v_str	
s4_y	$y_{s4}$	State	State in lag transfer function	v_str	
pfilt_y	$y_{P_{filt}}$	State	State in lag TF	v_str	
s5_y	$y_{s5}$	State	State in lag TF	v_str	
PIV_xi	$xi_{PIV}$	State	Integrator output	v_str	
Pord	$Pord$	AliasState	Alias of s5_y		
vp	$V_p$	Algeb	Sensed lower-capped voltage	v_str	
pfaref	$\Phi_{ref}$	Algeb	power factor angle ref	rad	v_str
Qref	$Q_{ref}$	Algeb	external Q ref	p.u.	v_str
Qcpf	$Q_{cpf}$	Algeb	Q calculated from P and power factor	p.u.	v_str
PFsel	$PFsel$	Algeb	Output of PFFLAG selector	v_str	
Qerr	$Q_{err}$	Algeb	Reactive power error	v_str	
PIQ_ys	$y_{SPIQ}$	Algeb	PI summation before limit	v_str	

continues on next page

Table 18 – continued from previous page

Name	Symbol	Type	Description	Unit	Properties
PIQ_y	$y_{PIQ}$	Algeb	PI output		v_str
Vsel_x	$x_{Vsel}$	Algeb	Value before limiter		v_str
Vsel_y	$y_{Vsel}$	Algeb	Output after limiter and post gain		v_str
Verr	$V_{err}$	Algeb	Voltage error ( $V_{ref0}$ )		v_str
dBV_y	$y_{dBV}$	Algeb	Deadband type 1 output		v_str
Iqinj	$I_{qinj}$	Algeb	Additional Iq signal during under- or over-voltage		v_str
wg	$\omega_g$	Algeb	Drive train generator speed		v_str
Pref	$P_{ref}$	Algeb	external P ref	<i>p.u.</i>	v_str
Psel	$P_{sel}$	Algeb	Output selection of PFLAG		v_str
VDL1_y	$y_{VDL1}$	Algeb	Output of piecewise		v_str
VDL2_y	$y_{VDL2}$	Algeb	Output of piecewise		v_str
Ipmax	$I_{pmax}$	Algeb	Upper limit on $I_{pcmd}$		v_str
Iqmax	$I_{qmax}$	Algeb	Upper limit on $I_{qcmd}$		v_str
PIV_ys	$y_{SPIV}$	Algeb	PI summation before limit		v_str
PIV_y	$y_{PIV}$	Algeb	PI output		v_str
Qsel	$Q_{sel}$	Algeb	Selection output of QFLAG		v_str
IpHL_x	$x_{IpHL}$	Algeb	Value before limiter		v_str
IpHL_y	$y_{IpHL}$	Algeb	Output after limiter and post gain		v_str
IqHL_x	$x_{IqHL}$	Algeb	Value before limiter		v_str
IqHL_y	$y_{IqHL}$	Algeb	Output after limiter and post gain		v_str
a	$\theta$	ExtAlgeb	Bus voltage angle		
v	$V_d$	ExtAlgeb	d-axis bus voltage magnitude		
Pe	$Pe$	ExtAlgeb	Retrieved Pe of RenGen		
Qe	$Qe$	ExtAlgeb	Retrieved Qe of RenGen		
Ipcmd	$Ipcmd$	ExtAlgeb	Retrieved $I_{pcmd}$ of RenGen		
Iqcmd	$Iqcmd$	ExtAlgeb	Retrieved $I_{qcmd}$ of RenGen		

## Initialization Equations

Name	Symbol	Type	Initial Value
s0_y	$y_{s0}$	State	$V_d$
S1_y	$y_{S_1}$	State	$Pe$
PIQ_xi	$x_{iPIQ}$	State	0.0
s4_y	$y_{s4}$	State	$\frac{P_{Fsel}}{V_p}$
pfilt_y	$y_{P_{filt}}$	State	$P_{ref}$
s5_y	$y_{s5}$	State	$P_{sel}$
PIV_xi	$x_{iPIV}$	State	$-I_{qcmd0}s_1^{SWV}$
Pord	$Pord$	AliasState	
vp	$V_p$	Algeb	$V_d z_i^{VLower} + 0.01 z_l^{VLower}$
pfaref	$\Phi_{ref}$	Algeb	$\Phi_{ref0}$
Qref	$Q_{ref}$	Algeb	$-q_{ref0}$

Table 19 – continued from previous page

Name	Symbol	Type	Initial Value
Qcpf	$Q_{cpf}$	Algeb	$Q_0$
PFsel	$PFsel$	Algeb	$Q_{cpf}s_1^{SW_{PF}} + Q_{ref}s_0^{SW_{PF}}$
Qerr	$Q_{err}$	Algeb	$PFselz_i^{PFlim} + Q_{max}z_u^{PFlim} + Q_{min}z_l^{PFlim} - Qe$
PIQ_ys	$y_{PIQ}$	Algeb	$K_{qp}Q_{errs}s_1^{SW_V}$
PIQ_y	$y_{PIQ}$	Algeb	$V_{max}z_u^{lim_{PIQ}} + V_{min}z_l^{lim_{PIQ}} + y_{PIQ}z_i^{lim_{PIQ}}$
Vsel_x	$x_{Vsel}$	Algeb	$s_0^{SW_V} (Q_{cpf}s_1^{SW_{PF}} + Q_{ref}s_0^{SW_{PF}} + V_{ref1}) + s_1^{SW_V} y_{PIQ}$
Vsel_y	$y_{Vsel}$	Algeb	$V_{max}z_u^{lim_{Vsel}} + V_{min}z_l^{lim_{Vsel}} + x_{Vsel}z_i^{lim_{Vsel}}$
Verr	$V_{err}$	Algeb	$V_{ref0} - y_{s0}$
dbV_y	$y_{dbV}$	Algeb	$1.0z_l^{db_{abV}}(V_{err} - d_{bd1}) + 1.0z_u^{db_{abV}}(V_{err} - d_{bd2})$
Iqinj	$I_{qinj}$	Algeb	$K_{qv}y_{dbV}z_{Vdip} + fThld(1 - z_{Vdip})(I_{qfrz}p_{Thld} + K_{qv}n_{Thld}y_{dbV})$
wg	$\omega_g$	Algeb	1.0
Pref	$P_{ref}$	Algeb	$\frac{P_0}{\omega_g}$
Psel	$P_{sel}$	Algeb	$\omega_g s_1^{SW_P} y_{Pfilt} + s_0^{SW_P} y_{Pfilt}$
VDL1_y	$y_{VDL1}$	Algeb	FixPiecewise(( $I_{q1}, V_{q1} \geq y_{s0}$ ), ( $I_{q1} + k_{Vq12}(-V_{q1} + y_{s0}), V_{q2} \geq y_{s0}$ ), ( $I_{q2} + k_{Vq23}(-V_{q2} + y_{s0}), V_{q3} \geq y_{s0}$ ))
VDL2_y	$y_{VDL2}$	Algeb	FixPiecewise(( $I_{p1}, V_{p1} \geq y_{s0}$ ), ( $I_{p1} + k_{Vp12}(-V_{p1} + y_{s0}), V_{p2} \geq y_{s0}$ ), ( $I_{p2} + k_{Vp23}(-V_{p2} + y_{s0}), V_{p3} \geq y_{s0}$ ))
Ipmx	$I_{pmx}$	Algeb	$(1 - fThld_2) \left( \sqrt{I_{pmx20,nn}^2 s_0^{SW_{PQ}}} + s_1^{SW_{PQ}} (z_{VDL2}(I_{maxr}(1 - VDL2c) + VDL2c) + VDL2c) \right)$
Iqmx	$I_{qmx}$	Algeb	$\sqrt{I_{qmx,nn}^2 s_1^{SW_{PQ}}} + s_0^{SW_{PQ}} (z_{VDL1}(I_{maxr}(1 - VDL1c) + VDL1c) + VDL1c) - 100000$
PIV_ys	$y_{SPIV}$	Algeb	$-I_{qcmd0}s_1^{SW_V} + K_{vp}s_1^{SW_V} (-s_0^{SW_V} y_{s0} + y_{Vsel})$
PIV_y	$y_{PIV}$	Algeb	$I_{qmax}z_u^{lim_{PIV}} + I_{qmin}z_l^{lim_{PIV}} + y_{SPIV}z_i^{lim_{PIV}}$
Qsel	$Q_{sel}$	Algeb	$s_0^{SW_V} y_{s4} + s_1^{SW_V} y_{PIV}$
IpHL_x	$x_{IpHL}$	Algeb	$\frac{y_{s5}}{V_p}$
IpHL_y	$y_{IpHL}$	Algeb	$I_{pmx}z_u^{lim_{IpHL}} + I_{pmin}z_l^{lim_{IpHL}} + x_{IpHL}z_i^{lim_{IpHL}}$
IqHL_x	$x_{IqHL}$	Algeb	$I_{qinj} + Q_{sel}$
IqHL_y	$y_{IqHL}$	Algeb	$I_{qmax}z_u^{lim_{IqHL}} + I_{qmin}z_l^{lim_{IqHL}} + x_{IqHL}z_i^{lim_{IqHL}}$
a	$\theta$	ExtAlgeb	
v	$V_d$	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	$y_{s0}$	State	$V_d - y_{s0}$	$T_{rv}$
S1_y	$y_{S1}$	State	$Pe - y_{S1}$	$T_p$
PIQ_xi	$xi_{PIQ}$	State	$K_{qi} (1 - z_{Vdip}) \left( Q_{err} s_1^{SW_V} + 2y_{PIQ} - 2y_{sPIQ} \right)$	
s4_y	$y_{s4}$	State	$(1 - z_{Vdip}) \left( \frac{P_{Fsel}}{V_p} - y_{s4} \right)$	$T_{iq}$
pfilt_y	$y_{Pfilt}$	State	$P_{ref} - y_{Pfilt}$	0.02
s5_y	$y_{s5}$	State	$(1 - z_{Vdip}) (P_{sel} - y_{s5})$	$T_{pord}$
PIV_xi	$xi_{PIV}$	State	$K_{vi} (1 - z_{Vdip}) \left( s_1^{SW_V} \left( -s_0^{SW_V} y_{s0} + y_{Vsel} \right) + 2y_{PIV} - 2y_{s1} \right)$	
Pord	$Pord$	AliasState	0	

## Algebraic Equations

Table 20 – continued from previous page

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
Qsel	$Q_{sel}$	Algeb	$-Q_{sel} + s_0^{SWV} y_{s4} + s_1^{SWV} y_{PIV}$
IpHL_x	$x_{IpHL}$	Algeb	$-x_{IpHL} + \frac{y_{s5}}{V_p}$
IpHL_y	$y_{IpHL}$	Algeb	$I_{pmax} z_u^{lim_{IpHL}} + I_{pmin} z_l^{lim_{IpHL}} + x_{IpHL} z_i^{lim_{IpHL}} - y_{IpHL}$
IqHL_x	$x_{IqHL}$	Algeb	$I_{qinj} + Q_{sel} - x_{IqHL}$
IqHL_y	$y_{IqHL}$	Algeb	$I_{qmax} z_u^{lim_{IqHL}} + I_{qmin} z_l^{lim_{IqHL}} + x_{IqHL} z_i^{lim_{IqHL}} - y_{IqHL}$
a	$\theta$	ExtAlgeb	0
v	$V_d$	ExtAlgeb	0
Pe	$Pe$	ExtAlgeb	0
Qe	$Qe$	ExtAlgeb	0
Ipcmd	$Ipcmd$	ExtAlgeb	$-Ipcmd_0 + y_{IpHL}$
Iqcmd	$Iqcmd$	ExtAlgeb	$-Iqcmd_0 - y_{IqHL}$

## Services

Name	Symbol	Equation	Type
Ipcmd0	$Ipcmd0$	$\frac{P_0}{V_d}$	ConstService
Iqcmd0	$Iqcmd0$	$-\frac{Q_0}{V_d}$	ConstService
pfaref0	$\Phi_{ref0}$	$\text{atan}_2(Q_0, P_0)$	ConstService
Volt_dip	$z_{Vdip}$	$1 - z_i^{V_{cmp}}$	VarService
qref0	$q_{ref0}$	$Iqcmd_0 s_0^{SWV} (V_d z_i^{VLower} + 0.01 z_l^{VLower}) + s_1^{SWV} (V_d - V_{ref1})$	ConstService
PIQ_flag	$z_{PIQ}^{flag}$	0	EventFlag
s4_flag	$z_{s4}^{flag}$	0	EventFlag
pThld	$pThld$	Indicator( $T_{hld} > 0$ )	ConstService
nThld	$nThld$	Indicator( $T_{hld} < 0$ )	ConstService
Thld_abs	$ Thld $	$ T_{hld} $	ConstService
fThld	$fThld$	0	ExtendedEvent
s5_flag	$z_{s5}^{flag}$	0	EventFlag
kVq12	$k_{Vq12}$	$\frac{-I_{q1}+I_{q2}}{-V_{q1}+V_{q2}}$	ConstService
kVq23	$k_{Vq23}$	$\frac{-I_{q2}+I_{q3}}{-V_{q2}+V_{q3}}$	ConstService
kVq34	$k_{Vq34}$	$\frac{-I_{q3}+I_{q4}}{-V_{q3}+V_{q4}}$	ConstService
zVDL1	$z_{VDL1}$	$I_{q1} \leq I_{q2} \wedge I_{q2} \leq I_{q3} \wedge I_{q3} \leq I_{q4} \wedge V_{q1} \leq V_{q2} \wedge V_{q2} \leq V_{q3} \wedge V_{q3} \leq V_{q4}$	ConstService
kVp12	$k_{Vp12}$	$\frac{-I_{p1}+I_{p2}}{-V_{p1}+V_{p2}}$	ConstService
kVp23	$k_{Vp23}$	$\frac{-I_{p2}+I_{p3}}{-V_{p2}+V_{p3}}$	ConstService
kVp34	$k_{Vp34}$	$\frac{-I_{p3}+I_{p4}}{-V_{p3}+V_{p4}}$	ConstService
zVDL2	$z_{VDL2}$	$I_{p1} \leq I_{p2} \wedge I_{p2} \leq I_{p3} \wedge I_{p3} \leq I_{p4} \wedge V_{p1} \leq V_{p2} \wedge V_{p2} \leq V_{p3} \wedge V_{p3} \leq V_{p4}$	ConstService
fThld2	$fThld2$	0	ExtendedEvent
VDL1c	$VDL1c$	$y_{VDL1} < I_{maxr}$	VarService
VDL2c	$VDL2c$	$y_{VDL2} < I_{maxr}$	VarService

continues on next page

Table 21 – continued from previous page

Name	Symbol	Equation	Type
Ipmax2sq0	$I_{pmax20,nn}^2$	FixPiecewise ( $(0, I_{max}^2 - Iqcmd_0^2 \leq 0), (I_{max}^2 - Iqcmd_0^2, \text{True})$ )	ConstService
Ipmax2sq	$I_{pmax2}^2$	FixPiecewise ( $(0, I_{max}^2 - y_{IqHL}^2 \leq 0), (I_{max}^2 - y_{IqHL}^2, \text{True})$ )	VarService
Ipmaxh	$I_{pmaxh}$	0	VarHold
Iqmax2sq0	$I_{qmax,nn}^2$	FixPiecewise ( $(0, I_{max}^2 - Ipcmd_0^2 \leq 0), (I_{max}^2 - Ipcmd_0^2, \text{True})$ )	ConstService
Iqmax2sq	$I_{qmax2}^2$	FixPiecewise ( $(0, I_{max}^2 - y_{IpHL}^2 \leq 0), (I_{max}^2 - y_{IpHL}^2, \text{True})$ )	VarService
Ipmin	$I_{pmin}$	0.0	ConstService
PIV_flag	$z_{PIV}^{flag}$	0	EventFlag

## 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	$lim_{Pfilt}$	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	$s_0$	Lag	Voltage filter
S1	$S_1$	Lag	Pe filter
PIQ	$PIQ$	PITrackAWFreeze	
Vsel	$V_{sel}$	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	$s_5$	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	$I_{pHL}$	GainLimiter	
IqHL	$I_{qHL}$	GainLimiter	

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	

## 5.23.2 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

continues on next page

Table 22 – continued from previous page

Name	Symbol	Description	Default	Unit	Properties
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 ( $\leq 0$ )	-0.020		
dbd2	$d_{bd2}$	Upper bound of the voltage deadband ( $\geq 0$ )	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		
QMin	$Q_{min}$	Lower limit for reactive power regulator	-999		
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		
dPmin	$d_{Pin}$	Power reference min. ramp rate ( $< 0$ )	-999		
PMAX	$P_{max}$	Max. active power limit $> 0$	999		
PMIN	$P_{min}$	Min. active power limit	0		
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		

continues on next page

Table 22 – continued from previous page

Name	Symbol	Description	Default	Unit	Properties
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			
bus		Retrieved bus idx			
gen		Retrieved StaticGen idx			
Sn	$S_n$		0		

## Variables

Name	Symbol	Type	Description	Unit	Properties
s0_y	$y_{s0}$	State	State in lag transfer function		v_str
S1_y	$y_{S_1}$	State	State in lag transfer function		v_str
PIQ_xi	$x_{iPIQ}$	State	Integrator output		v_str
s4_y	$y_{s4}$	State	State in lag transfer function		v_str
pfilt_y	$y_{P_{filt}}$	State	State in lag TF		v_str
s5_y	$y_{s5}$	State	State in lag TF		v_str
PIV_xi	$x_{iPIV}$	State	Integrator output		v_str
Pord	$Pord$	AliasState	Alias of s5_y		
vp	$V_p$	Algeb	Sensed lower-capped voltage		v_str
pfaref	$\Phi_{ref}$	Algeb	power factor angle ref	rad	v_str
Qref	$Q_{ref}$	Algeb	external Q ref	p.u.	v_str
Qcpf	$Q_{cpf}$	Algeb	Q calculated from P and power factor	p.u.	v_str
PFsel	$PFsel$	Algeb	Output of PFFLAG selector		v_str
Qerr	$Q_{err}$	Algeb	Reactive power error		v_str
PIQ_ys	$y_{sPIQ}$	Algeb	PI summation before limit		v_str
PIQ_y	$y_{PIQ}$	Algeb	PI output		v_str
Vsel_x	$x_{Vsel}$	Algeb	Value before limiter		v_str
Vsel_y	$y_{Vsel}$	Algeb	Output after limiter and post gain		v_str
Verr	$V_{err}$	Algeb	Voltage error (Vref0)		v_str
dbV_y	$y_{dbV}$	Algeb	Deadband type 1 output		v_str
Iqinj	$I_{qinj}$	Algeb	Additional Iq signal during under- or over-voltage		v_str
wg	$\omega_g$	Algeb	Drive train generator speed		v_str
Pref	$P_{ref}$	Algeb	external P ref	p.u.	v_str
Psel	$P_{sel}$	Algeb	Output selection of PFLAG		v_str

continues on next page

Table 23 – continued from previous page

Name	Symbol	Type	Description	Unit	Properties
VDL1_y	$y_{VDL1}$	Algeb	Output of piecewise	v_str	
VDL2_y	$y_{VDL2}$	Algeb	Output of piecewise	v_str	
Ipmax	$I_{pmax}$	Algeb	Upper limit on Ipcmd	v_str	
Iqmax	$I_{qmax}$	Algeb	Upper limit on Iqcmd	v_str	
PIV_ys	$y_{SPIV}$	Algeb	PI summation before limit	v_str	
PIV_y	$y_{PIV}$	Algeb	PI output	v_str	
Qsel	$Q_{sel}$	Algeb	Selection output of QFLAG	v_str	
IpHL_x	$x_{IpHL}$	Algeb	Value before limiter	v_str	
IpHL_y	$y_{IpHL}$	Algeb	Output after limiter and post gain	v_str	
IqHL_x	$x_{IqHL}$	Algeb	Value before limiter	v_str	
IqHL_y	$y_{IqHL}$	Algeb	Output after limiter and post gain	v_str	
a	$\theta$	ExtAlgeb	Bus voltage angle		
v	$V_d$	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	$df dt$	ExtAlgeb	Bus ROCOF	p.u.	

## Initialization Equations

Name	Symbol	Type	Initial Value
s0_y	$y_{s0}$	State	$V_d$
S1_y	$y_{S1}$	State	$Pe$
PIQ_xi	$xi_{PIQ}$	State	0.0
s4_y	$y_{s4}$	State	$\frac{PF_{sel}}{V_p}$
pfilt_y	$y_{Pfilt}$	State	$P_{ref}$
s5_y	$y_{s5}$	State	$P_{sel}$
PIV_xi	$xi_{PIV}$	State	$-Iqcmd_0 s_1^{SW_V}$
Pord	$Pord$	AliasState	
vp	$V_p$	Algeb	$V_d z_i^{VLower} + 0.01 z_l^{VLower}$
pfaref	$\Phi_{ref}$	Algeb	$\Phi_{ref0}$
Qref	$Q_{ref}$	Algeb	$-q_{ref0}$
Qcpf	$Q_{cpf}$	Algeb	$Q_0$
PFsel	$PFsel$	Algeb	$Q_{cpf} s_1^{SW_{PF}} + Q_{ref} s_0^{SW_{PF}}$
Qerr	$Q_{err}$	Algeb	$PFsel z_i^{PFlim} + Q_{max} z_u^{PFlim} + Q_{min} z_l^{PFlim} - Qe$
PIQ_ys	$y_{SPIQ}$	Algeb	$K_{qp} Q_{err} s_1^{SW_V}$
PIQ_y	$y_{PIQ}$	Algeb	$V_{max} z_u^{lim_{PIQ}} + V_{min} z_l^{lim_{PIQ}} + y_{SPIQ} z_i^{lim_{PIQ}}$
Vsel_x	$x_{Vsel}$	Algeb	$s_0^{SW_V} \left( Q_{cpf} s_1^{SW_{PF}} + Q_{ref} s_0^{SW_{PF}} + V_{ref1} \right) + s_1^{SW_V} y_{PIQ}$

Table 24 – continued from previous page

Name	Symbol	Type	Initial Value
Vsel_y	$y_{Vsel}$	Algeb	$V_{max}z_u^{lim_{Vsel}} + V_{min}z_l^{lim_{Vsel}} + x_{Vsel}z_i^{lim_{Vsel}}$
Verr	$V_{err}$	Algeb	$V_{ref0} - y_{s0}$
dbV_y	$y_{dbV}$	Algeb	$1.0z_l^{db_{dbV}}(V_{err} - d_{bd1}) + 1.0z_u^{db_{dbV}}(V_{err} - d_{bd2})$
Iqinj	$I_{qinj}$	Algeb	$K_{qv}y_{dbV}z_{Vdip} + fThld(1 - z_{Vdip})(I_{qfrz}p_{Thld} + K_{qv}n_{Thld}y_{dbV})$
wg	$\omega_g$	Algeb	1.0
Pref	$P_{ref}$	Algeb	$\frac{P_0}{\omega_g}$
Psel	$P_{sel}$	Algeb	$\omega_g s_1^{SW_P} y_{Pfilt} + s_0^{SW_P} y_{Pfilt}$
VDL1_y	$y_{VDL1}$	Algeb	FixPiecewise(( $I_{q1}, V_{q1} \geq y_{s0}$ ), ( $I_{q1} + k_{Vq12}(-V_{q1} + y_{s0}), V_{q2} \geq y_{s0}$ ), ( $I_{q2} + k_{Vq23}(-V_{q2} + y_{s0}), V_{q3} \geq y_{s0}$ ))
VDL2_y	$y_{VDL2}$	Algeb	FixPiecewise(( $I_{p1}, V_{p1} \geq y_{s0}$ ), ( $I_{p1} + k_{Vp12}(-V_{p1} + y_{s0}), V_{p2} \geq y_{s0}$ ), ( $I_{p2} + k_{Vp23}(-V_{p2} + y_{s0}), V_{p3} \geq y_{s0}$ ))
Ipmax	$I_{pmax}$	Algeb	$(1 - fThld_2) \left( \sqrt{I_{pmax20,nn}^2 s_0^{SW_{PQ}}} + s_1^{SW_{PQ}} (z_{VDL2}(I_{maxr}(1 - VDL2c) + VDL2c) + VDL2c) \right)$
Iqmax	$I_{qmax}$	Algeb	$\sqrt{I_{qmax,nn}^2 s_1^{SW_{PQ}}} + s_0^{SW_{PQ}} (z_{VDL1}(I_{maxr}(1 - VDL1c) + VDL1c) + VDL1c) - 100000$
PIV_ys	$y_{SPIV}$	Algeb	$-I_{qcmand} s_1^{SW_V} + K_{vp} s_1^{SW_V} (-s_0^{SW_V} y_{s0} + y_{Vsel})$
PIV_y	$y_{PIV}$	Algeb	$I_{qmax}z_u^{lim_{PIV}} + I_{qmin}z_l^{lim_{PIV}} + y_{SPIV}z_i^{lim_{PIV}}$
Qsel	$Q_{sel}$	Algeb	$s_0^{SW_V} y_{s4} + s_1^{SW_V} y_{PIV}$
IpHL_x	$x_{IpHL}$	Algeb	$\frac{y_{s5}}{V_p}$
IpHL_y	$y_{IpHL}$	Algeb	$I_{pmax}z_u^{lim_{IpHL}} + I_{pmin}z_l^{lim_{IpHL}} + x_{IpHL}z_i^{lim_{IpHL}}$
IqHL_x	$x_{IqHL}$	Algeb	$I_{qinj} + Q_{sel}$
IqHL_y	$y_{IqHL}$	Algeb	$I_{qmax}z_u^{lim_{IqHL}} + I_{qmin}z_l^{lim_{IqHL}} + x_{IqHL}z_i^{lim_{IqHL}}$
a	$\theta$	ExtAlgeb	
v	$V_d$	ExtAlgeb	
Pe	$Pe$	ExtAlgeb	
Qe	$Qe$	ExtAlgeb	
Ipcmd	$Ipcmd$	ExtAlgeb	
Iqcmd	$Iqcmd$	ExtAlgeb	
df	$df$	ExtAlgeb	
dfdt	$dfdt$	ExtAlgeb	

## Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
s0_y	$y_{s0}$	State	$V_d - y_{s0}$	$T_{rv}$
S1_y	$y_{S1}$	State	$P_e - y_{S1}$	$T_p$
PIQ_xi	$xi_{PIQ}$	State	$K_{qi} (1 - z_{Vdip}) \left( Q_{err} s_1^{SW_V} + 2y_{PIQ} - 2y_{sPIQ} \right)$	
s4_y	$y_{s4}$	State	$(1 - z_{Vdip}) \left( \frac{PFsel}{V_p} - y_{s4} \right)$	$T_{iq}$
pfilt_y	$y_{Pfilt}$	State	$P_{ref} - y_{Pfilt}$	0.02
s5_y	$y_{s5}$	State	$(1 - z_{Vdip}) (P_{sel} - y_{s5})$	$T_{pord}$
PIV_xi	$xi_{PIV}$	State	$K_{vi} (1 - z_{Vdip}) \left( s_1^{SW_V} \left( -s_0^{SW_V} y_{s0} + y_{Vsel} \right) + 2y_{PIV} - 2y_{sI} \right)$	
Pord	$Pord$	AliasState	0	

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
vp	$V_p$	Algeb	$V_d z_i^{VLower} - V_p + 0.01 z_l^{VLower}$
pfaref	$\Phi_{ref}$	Algeb	$\Phi_{ref0} - \Phi_{ref}$
Qref	$Q_{ref}$	Algeb	$-Q_{ref} - q_{ref0}$
Qcpf	$Q_{cpf}$	Algeb	$(1 - z_1^{zp}) (-Q_{cpf} + y_{S1} \tan(\Phi_{ref}))$
PFsel	$PFsel$	Algeb	$-PFsel + Q_{cpf} s_1^{SW_{PF}} + Q_{ref} s_0^{SW_{PF}}$
Qerr	$Q_{err}$	Algeb	$PFsel z_i^{PFlim} - Q_{err} + Q_{max} z_u^{PFlim} + Q_{min} z_l^{PFlim} - Q_e$
PIQ_ys	$y_{sPIQ}$	Algeb	$(1 - z_{Vdip}) \left( K_{qp} Q_{err} s_1^{SW_V} + xi_{PIQ} - y_{sPIQ} \right)$
PIQ_y	$y_{PIQ}$	Algeb	$(1 - z_{Vdip}) \left( V_{max} z_u^{lim_{PIQ}} + V_{min} z_l^{lim_{PIQ}} - y_{PIQ} + y_{sPIQ} z_i^{lim_{PIQ}} \right)$
Vsel_x	$x_{Vsel}$	Algeb	$s_0^{SW_V} \left( Q_{cpf} s_1^{SW_{PF}} + Q_{ref} s_0^{SW_{PF}} + V_{ref1} \right) + s_1^{SW_V} y_{PIQ} - x_{Vsel}$
Vsel_y	$y_{Vsel}$	Algeb	$V_{max} z_u^{lim_{Vsel}} + V_{min} z_l^{lim_{Vsel}} + x_{Vsel} z_i^{lim_{Vsel}} - y_{Vsel}$
Verr	$V_{err}$	Algeb	$-V_{err} + V_{ref0} - y_{s0}$
dbV_y	$y_{dbV}$	Algeb	$-y_{dbV} + 1.0 z_l^{db_{dbV}} (V_{err} - db_{d1}) + 1.0 z_u^{db_{dbV}} (V_{err} - db_{d2})$
Iqinj	$I_{qinj}$	Algeb	$-I_{qinj} + K_{qv} y_{dbV} z_{Vdip} + fThld (1 - z_{Vdip}) (I_{qfrz} p_{Thld} + K_{qv} n_{Thld} y_{dbV})$
wg	$\omega_g$	Algeb	$1.0 - \omega_g$
Pref	$P_{ref}$	Algeb	$-K_{df} df - K_{df} df dt + \frac{P_0}{\omega_g} - P_{ref}$
Psel	$P_{sel}$	Algeb	$-P_{sel} + \omega_g s_1^{SW_P} y_{Pfilt} + s_0^{SW_P} y_{Pfilt}$
VDL1_y	$y_{V_{DL1}}$	Algeb	$-y_{V_{DL1}} + \text{FixPiecewise}((I_{q1}, V_{q1} \geq y_{s0}), (I_{q1} + k_{Vq12} (-V_{q1} + y_{s0}), V_{q2} \geq y_{s0}), (I_{q1} + k_{Vq12} (-V_{q1} + y_{s0}), V_{q2} < y_{s0}), (I_{q1} + k_{Vq12} (-V_{q1} + y_{s0}), V_{q2} \geq y_{s0}), (I_{q1} + k_{Vq12} (-V_{q1} + y_{s0}), V_{q2} < y_{s0}))$
VDL2_y	$y_{V_{DL2}}$	Algeb	$-y_{V_{DL2}} + \text{FixPiecewise}((I_{p1}, V_{p1} \geq y_{s0}), (I_{p1} + k_{Vp12} (-V_{p1} + y_{s0}), V_{p2} \geq y_{s0}), (I_{p1} + k_{Vp12} (-V_{p1} + y_{s0}), V_{p2} < y_{s0}), (I_{p1} + k_{Vp12} (-V_{p1} + y_{s0}), V_{p2} \geq y_{s0}), (I_{p1} + k_{Vp12} (-V_{p1} + y_{s0}), V_{p2} < y_{s0}))$
Ipmax	$I_{pmax}$	Algeb	$-I_{pmax} + I_{pmax} h_{Thld2} + (1 - f_{Thld2}) \left( \sqrt{I_{pmax2}^2 s_0^{SW_{PQ}}} + s_1^{SW_{PQ}} (z_{V_{DL2}} (I_{max} - I_{pmax}) + I_{pmax} (I_{max} - I_{pmax})) \right)$
Iqmax	$I_{qmax}$	Algeb	$\sqrt{I_{qmax2}^2 s_1^{SW_{PQ}}} - I_{qmax} + s_0^{SW_{PQ}} (z_{V_{DL1}} (I_{maxr} (1 - V_{DL1c}) + V_{DL1c} y_{V_{DL1}}) - I_{qmax} (I_{maxr} (1 - V_{DL1c}) + V_{DL1c} y_{V_{DL1}}))$
PIV_ys	$y_{sPIV}$	Algeb	$(1 - z_{Vdip}) \left( K_{vp} s_1^{SW_V} \left( -s_0^{SW_V} y_{s0} + y_{Vsel} \right) + xi_{PIV} - y_{sPIV} \right)$
PIV_y	$y_{PIV}$	Algeb	$(1 - z_{Vdip}) \left( I_{qmax} z_u^{lim_{PIV}} + I_{qmin} z_l^{lim_{PIV}} - y_{PIV} + y_{sPIV} z_i^{lim_{PIV}} \right)$

Table 25 – continued from previous page

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
Qsel	$Q_{sel}$	Algeb	$-Q_{sel} + s_0^{SWV} y_{s4} + s_1^{SWV} y_{PIV}$
IpHL_x	$x_{IpHL}$	Algeb	$-x_{IpHL} + \frac{y_{s5}}{V_p}$
IpHL_y	$y_{IpHL}$	Algeb	$I_{pmax} z_u^{lim_{IpHL}} + I_{pmin} z_l^{lim_{IpHL}} + x_{IpHL} z_i^{lim_{IpHL}} - y_{IpHL}$
IqHL_x	$x_{IqHL}$	Algeb	$I_{qinj} + Q_{sel} - x_{IqHL}$
IqHL_y	$y_{IqHL}$	Algeb	$I_{qmax} z_u^{lim_{IqHL}} + I_{qmin} z_l^{lim_{IqHL}} + x_{IqHL} z_i^{lim_{IqHL}} - y_{IqHL}$
a	$\theta$	ExtAlgeb	0
v	$V_d$	ExtAlgeb	0
Pe	$Pe$	ExtAlgeb	0
Qe	$Qe$	ExtAlgeb	0
Ipcmd	$Ipcmd$	ExtAlgeb	$-Ipcmd_0 + y_{IpHL}$
Iqcmd	$Iqcmd$	ExtAlgeb	$-Iqcmd_0 - y_{IqHL}$
df	$df$	ExtAlgeb	0
dfdt	$dfdt$	ExtAlgeb	0

## Services

Name	Symbol	Equation	Type
Ipcmd0	$Ipcmd0$	$\frac{P_0}{V_d}$	ConstService
Iqcmd0	$Iqcmd0$	$-\frac{Q_0}{V_d}$	ConstService
pfaref0	$\Phi_{ref0}$	$\text{atan}_2(Q_0, P_0)$	ConstService
Volt_dip	$z_{Vdip}$	$1 - z_i^{V_{cmp}}$	VarService
qref0	$q_{ref0}$	$Iqcmd_0 s_0^{SWV} (V_d z_i^{VLower} + 0.01 z_l^{VLower}) + s_1^{SWV} (V_d - V_{ref1})$	ConstService
PIQ_flag	$z_{PIQ}^{flag}$	0	EventFlag
s4_flag	$z_{s4}^{flag}$	0	EventFlag
pThld	$pThld$	Indicator( $T_{hld} > 0$ )	ConstService
nThld	$nThld$	Indicator( $T_{hld} < 0$ )	ConstService
Thld_abs	$ T_{hld} $	$ T_{hld} $	ConstService
fThld	$fThld$	0	ExtendedEvent
s5_flag	$z_{s5}^{flag}$	0	EventFlag
kVq12	$k_{Vq12}$	$\frac{-I_{q1}+I_{q2}}{-V_{q1}+V_{q2}}$	ConstService
kVq23	$k_{Vq23}$	$\frac{-I_{q2}+I_{q3}}{-V_{q2}+V_{q3}}$	ConstService
kVq34	$k_{Vq34}$	$\frac{-I_{q3}+I_{q4}}{-V_{q3}+V_{q4}}$	ConstService
zVDL1	$z_{VDL1}$	$I_{q1} \leq I_{q2} \wedge I_{q2} \leq I_{q3} \wedge I_{q3} \leq I_{q4} \wedge V_{q1} \leq V_{q2} \wedge V_{q2} \leq V_{q3} \wedge V_{q3} \leq V_{q4}$	ConstService
kVp12	$k_{Vp12}$	$\frac{-I_{p1}+I_{p2}}{-V_{p1}+V_{p2}}$	ConstService
kVp23	$k_{Vp23}$	$\frac{-I_{p2}+I_{p3}}{-V_{p2}+V_{p3}}$	ConstService
kVp34	$k_{Vp34}$	$\frac{-I_{p3}+I_{p4}}{-V_{p3}+V_{p4}}$	ConstService
zVDL2	$z_{VDL2}$	$I_{p1} \leq I_{p2} \wedge I_{p2} \leq I_{p3} \wedge I_{p3} \leq I_{p4} \wedge V_{p1} \leq V_{p2} \wedge V_{p2} \leq V_{p3} \wedge V_{p3} \leq V_{p4}$	ConstService
fThld2	$fThld2$	0	ExtendedEvent

continues on next page

Table 26 – continued from previous page

Name	Symbol	Equation	Type
VDL1c	$VDL1c$	$y_{VDL1} < I_{maxr}$	VarService
VDL2c	$VDL2c$	$y_{VDL2} < I_{maxr}$	VarService
Ipmax2sq0	$I_{pmax20,nn}^2$	FixPiecewise $((0, I_{max}^2 - I_{qcmd0}^2 \leq 0), (I_{max}^2 - I_{qcmd0}^2, \text{True}))$	ConstService
Ipmax2sq	$I_{pmax2}^2$	FixPiecewise $((0, I_{max}^2 - y_{IqHL}^2 \leq 0), (I_{max}^2 - y_{IqHL}^2, \text{True}))$	VarService
Ipmaxh	$Ipmaxh$	0	VarHold
Iqmax2sq0	$I_{qmax,nn}^2$	FixPiecewise $((0, I_{max}^2 - I_{pcmd0}^2 \leq 0), (I_{max}^2 - I_{pcmd0}^2, \text{True}))$	ConstService
Iqmax2sq	$I_{qmax2}^2$	FixPiecewise $((0, I_{max}^2 - y_{IpHL}^2 \leq 0), (I_{max}^2 - y_{IpHL}^2, \text{True}))$	VarService
Ipmin	$I_{pmin}$	0.0	ConstService
PIV_flag	$z_{PIV}^{flag}$	0	EventFlag

## Discretes

Name	Symbol	Type	Info
SWPF	$SW_{PF}$	Switcher	
SWV	$SW_V$	Switcher	
SWQ	$SW_Q$	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	$lim_{Pfilt}$	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	$s_0$	Lag	Voltage filter
S1	$S_1$	Lag	Pe filter
PIQ	$PIQ$	PITrackAWFreeze	
Vsel	$V_{sel}$	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	$s_5$	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	$I_{pHL}$	GainLimiter	
IqHL	$I_{qHL}$	GainLimiter	

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	

### 5.23.3 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		
QMin	$Q_{min}$	Lower limit for reactive power regulator	-999		
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	$dP_{max}$	Power reference max. ramp rate ( $>0$ )	999		
dPmin	$dP_{min}$	Power reference min. ramp rate ( $<0$ )	-999		
PMAX	$P_{max}$	Max. active power limit $> 0$	999		
PMIN	$P_{min}$	Min. active power limit	0		
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		

continues on next page

Table 27 – continued from previous page

Name	Symbol	Description	Default	Unit	Properties
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
bus		Retrieved bus idx			
gen		Retrieved StaticGen idx			
Sn	$S_n$		0		

## Variables

Name	Symbol	Type	Description	Unit	Properties
s0_y	$y_{s0}$	State	State in lag transfer function		v_str
S1_y	$y_{S_1}$	State	State in lag transfer function		v_str
PIQ_xi	$x_{iPIQ}$	State	Integrator output		v_str
s4_y	$y_{s4}$	State	State in lag transfer function		v_str
pfilt_y	$y_{P_{filt}}$	State	State in lag TF		v_str
s5_y	$y_{s5}$	State	State in lag TF		v_str
PIV_xi	$x_{iPIV}$	State	Integrator output		v_str
Pord	$Pord$	AliasState	Alias of s5_y		
omega	$\omega$	ExtState	generator speed	pu	
vp	$V_p$	Algeb	Sensed lower-capped voltage		v_str
pfaref	$\Phi_{ref}$	Algeb	power factor angle ref	rad	v_str
Qref	$Q_{ref}$	Algeb	external Q ref	pu.	v_str
Qcpf	$Q_{cpf}$	Algeb	Q calculated from P and power factor	pu.	v_str
PFsel	$PFsel$	Algeb	Output of PFFLAG selector		v_str
Qerr	$Q_{err}$	Algeb	Reactive power error		v_str
PIQ_ys	$y_{sPIQ}$	Algeb	PI summation before limit		v_str
PIQ_y	$y_{PIQ}$	Algeb	PI output		v_str
Vsel_x	$x_{Vsel}$	Algeb	Value before limiter		v_str
Vsel_y	$y_{Vsel}$	Algeb	Output after limiter and post gain		v_str

continues on next page

Table 28 – continued from previous page

Name	Symbol	Type	Description	Unit	Properties
Verr	$V_{err}$	Algeb	Voltage error ( $V_{ref0}$ )		v_str
dbV_y	$y_{dbV}$	Algeb	Deadband type 1 output		v_str
Iqinj	$I_{qinj}$	Algeb	Additional Iq signal during under- or over-voltage		v_str
wg	$\omega_g$	Algeb	Drive train generator speed		v_str
Pref	$P_{ref}$	Algeb	external P ref	<i>p.u.</i>	v_str
Psel	$P_{sel}$	Algeb	Output selection of PFLAG		v_str
VDL1_y	$y_{VDL1}$	Algeb	Output of piecewise		v_str
VDL2_y	$y_{VDL2}$	Algeb	Output of piecewise		v_str
Ipmax	$I_{pmax}$	Algeb	Upper limit on Ipcmd		v_str
Iqmax	$I_{qmax}$	Algeb	Upper limit on Iqcmd		v_str
PIV_ys	$y_{SPIV}$	Algeb	PI summation before limit		v_str
PIV_y	$y_{PIV}$	Algeb	PI output		v_str
Qsel	$Q_{sel}$	Algeb	Selection output of QFLAG		v_str
IpHL_x	$x_{IpHL}$	Algeb	Value before limiter		v_str
IpHL_y	$y_{IpHL}$	Algeb	Output after limiter and post gain		v_str
IqHL_x	$x_{IqHL}$	Algeb	Value before limiter		v_str
IqHL_y	$y_{IqHL}$	Algeb	Output after limiter and post gain		v_str
a	$\theta$	ExtAlgeb	Bus voltage angle		
v	$V_d$	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	$y_{s0}$	State	$V_d$
S1_y	$y_{S1}$	State	$Pe$
PIQ_xi	$x_{iPIQ}$	State	0.0
s4_y	$y_{s4}$	State	$\frac{PFsel}{V_p}$
pfilt_y	$y_{Pfilt}$	State	$P_{ref}$
s5_y	$y_{s5}$	State	$P_{sel}$
PIV_xi	$x_{iPIV}$	State	$-Iqcmd_0 s_1^{SW_V}$
Pord	$Pord$	AliasState	
omega	$\omega$	ExtState	
vp	$V_p$	Algeb	$V_d z_i^{VLower} + 0.01 z_l^{VLower}$
pfaref	$\Phi_{ref}$	Algeb	$\Phi_{ref0}$
Qref	$Q_{ref}$	Algeb	$-q_{ref0}$
Qcpf	$Q_{cpf}$	Algeb	$Q_0$
PFsel	$PFsel$	Algeb	$Q_{cpf} s_1^{SW_{PF}} + Q_{ref} s_0^{SW_{PF}}$

Table 29 – continued from previous pa

Name	Symbol	Type	Initial Value
Qerr	$Q_{err}$	Algeb	$PFselz_i^{PFlim} + Q_{max}z_u^{PFlim} + Q_{min}z_l^{PFlim} - Qe$
PIQ_ys	$y_{SPIQ}$	Algeb	$K_{qp}Q_{err}s_1^{SW_V}$
PIQ_y	$y_{PIQ}$	Algeb	$V_{max}z_u^{lim_{PIQ}} + V_{min}z_l^{lim_{PIQ}} + y_{SPIQ}z_i^{lim_{PIQ}}$
Vsel_x	$x_{Vsel}$	Algeb	$s_0^{SW_V} \left( Q_{cpf}s_1^{SW_{PF}} + Q_{ref}s_0^{SW_{PF}} + V_{ref1} \right) + s_1^{SW_V} y_{PIQ}$
Vsel_y	$y_{Vsel}$	Algeb	$V_{max}z_u^{lim_{Vsel}} + V_{min}z_l^{lim_{Vsel}} + x_{Vsel}z_i^{lim_{Vsel}}$
Verr	$V_{err}$	Algeb	$V_{ref0} - y_{s0}$
dbV_y	$y_{dbV}$	Algeb	$1.0z_l^{db_{dbV}} (V_{err} - d_{bd1}) + 1.0z_u^{db_{dbV}} (V_{err} - d_{bd2})$
Iqinj	$I_{qinj}$	Algeb	$K_{qv}y_{dbV}z_{Vdip} + fThld(1 - z_{Vdip})(I_{qfrz}p_{Thld} + K_{qv}n_{Thld}y_{dbV})$
wg	$\omega_g$	Algeb	1.0
Pref	$P_{ref}$	Algeb	$\frac{P_0}{\omega_g}$
Psel	$P_{sel}$	Algeb	$\omega_g s_1^{SW_P} y_{Pfilt} + s_0^{SW_P} y_{Pfilt}$
VDL1_y	$y_{VDL1}$	Algeb	FixPiecewise(( $I_{q1}, V_{q1} \geq y_{s0}$ ), ( $I_{q1} + k_{Vq12}(-V_{q1} + y_{s0}), V_{q2} \geq y_{s0}$ ), ( $I_{q2} + k_{Vq23}(-V_{q2} + y_{s0}), V_{q3} \geq y_{s0}$ ))
VDL2_y	$y_{VDL2}$	Algeb	FixPiecewise(( $I_{p1}, V_{p1} \geq y_{s0}$ ), ( $I_{p1} + k_{Vp12}(-V_{p1} + y_{s0}), V_{p2} \geq y_{s0}$ ), ( $I_{p2} + k_{Vp23}(-V_{p2} + y_{s0}), V_{p3} \geq y_{s0}$ ))
Ipmax	$I_{pmax}$	Algeb	$(1 - fThld_2) \left( \sqrt{I_{pmax20,nn}^2 s_0^{SW_{PQ}}} + s_1^{SW_{PQ}} (z_{VDL1}(I_{maxr}(1 - VDL2c) + VDL2c) - 100000) \right)$
Iqmax	$I_{qmax}$	Algeb	$\sqrt{I_{qmax,nn}^2 s_1^{SW_{PQ}}} + s_0^{SW_{PQ}} (z_{VDL1}(I_{maxr}(1 - VDL1c) + VDL1c) - 100000)$
PIV_ys	$y_{SPIV}$	Algeb	$-Iqcmd_0 s_1^{SW_V} + K_{vp}s_1^{SW_V} (-s_0^{SW_V} y_{s0} + y_{Vsel})$
PIV_y	$y_{PIV}$	Algeb	$I_{qmax}z_u^{lim_{PIV}} + I_{qmin}z_l^{lim_{PIV}} + y_{SPIV}z_i^{lim_{PIV}}$
Qsel	$Q_{sel}$	Algeb	$s_0^{SW_V} y_{s4} + s_1^{SW_V} y_{PIV}$
IpHL_x	$x_{IpHL}$	Algeb	$\frac{y_{s5}}{V_p}$
IpHL_y	$y_{IpHL}$	Algeb	$I_{pmax}z_u^{lim_{IpHL}} + I_{pmin}z_l^{lim_{IpHL}} + x_{IpHL}z_i^{lim_{IpHL}}$
IqHL_x	$x_{IqHL}$	Algeb	$I_{qinj} + Q_{sel}$
IqHL_y	$y_{IqHL}$	Algeb	$I_{qmax}z_u^{lim_{IqHL}} + I_{qmin}z_l^{lim_{IqHL}} + x_{IqHL}z_i^{lim_{IqHL}}$
a	$\theta$	ExtAlgeb	
v	$V_d$	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	$y_{s0}$	State	$V_d - y_{s0}$	$T_{rv}$
S1_y	$y_{S1}$	State	$P_e - y_{S1}$	$T_p$
PIQ_xi	$xi_{PIQ}$	State	$K_{qi} (1 - z_{Vdip}) \left( Q_{err} s_1^{SW_V} + 2y_{PIQ} - 2y_{sPIQ} \right)$	
s4_y	$y_{s4}$	State	$(1 - z_{Vdip}) \left( \frac{PFsel}{V_p} - y_{s4} \right)$	$T_{iq}$
pfilt_y	$y_{Pfilt}$	State	$P_{ref} - y_{Pfilt}$	0.02
s5_y	$y_{s5}$	State	$(1 - z_{Vdip}) (P_{sel} - y_{s5})$	$T_{pord}$
PIV_xi	$xi_{PIV}$	State	$K_{vi} (1 - z_{Vdip}) \left( s_1^{SW_V} \left( -s_0^{SW_V} y_{s0} + y_{Vsel} \right) + 2y_{PIV} - 2y_{sI} \right)$	
Pord	$Pord$	AliasState	0	
omega	$\omega$	ExtState	0	

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
vp	$V_p$	Algeb	$V_d z_i^{VLower} - V_p + 0.01 z_l^{VLower}$
pfaref	$\Phi_{ref}$	Algeb	$\Phi_{ref0} - \Phi_{ref}$
Qref	$Q_{ref}$	Algeb	$-Q_{ref} - q_{ref0}$
Qcpf	$Q_{cpf}$	Algeb	$(1 - z_1^{zp}) (-Q_{cpf} + y_{S1} \tan(\Phi_{ref}))$
PFsel	$PFsel$	Algeb	$-PFsel + Q_{cpf} s_1^{SW_{PF}} + Q_{ref} s_0^{SW_{PF}}$
Qerr	$Q_{err}$	Algeb	$PFsel z_i^{PFlim} - Q_{err} + Q_{max} z_u^{PFlim} + Q_{min} z_l^{PFlim} - Q_e$
PIQ_ys	$y_{sPIQ}$	Algeb	$(1 - z_{Vdip}) \left( K_{qp} Q_{err} s_1^{SW_V} + xi_{PIQ} - y_{sPIQ} \right)$
PIQ_y	$y_{PIQ}$	Algeb	$(1 - z_{Vdip}) \left( V_{max} z_u^{lim_{PIQ}} + V_{min} z_l^{lim_{PIQ}} - y_{PIQ} + y_{sPIQ} z_i^{lim_{PIQ}} \right)$
Vsel_x	$x_{Vsel}$	Algeb	$s_0^{SW_V} \left( Q_{cpf} s_1^{SW_{PF}} + Q_{ref} s_0^{SW_{PF}} + V_{ref1} \right) + s_1^{SW_V} y_{PIQ} - x_{Vsel}$
Vsel_y	$y_{Vsel}$	Algeb	$V_{max} z_u^{lim_{Vsel}} + V_{min} z_l^{lim_{Vsel}} + x_{Vsel} z_i^{lim_{Vsel}} - y_{Vsel}$
Verr	$V_{err}$	Algeb	$-V_{err} + V_{ref0} - y_{s0}$
dbV_y	$y_{dbV}$	Algeb	$-y_{dbV} + 1.0 z_l^{db_{dbV}} (V_{err} - d_{bd1}) + 1.0 z_u^{db_{dbV}} (V_{err} - d_{bd2})$
Iqinj	$I_{qinj}$	Algeb	$-I_{qinj} + K_{qv} y_{dbV} z_{Vdip} + fThld (1 - z_{Vdip}) (I_{qfrz} pThld + K_{qv} nThld y_{dbV})$
wg	$\omega_g$	Algeb	$1.0 - \omega_g$
Pref	$P_{ref}$	Algeb	$-K_{df} (\omega - 1) + \frac{P_0}{\omega_g} - P_{ref}$
Psel	$P_{sel}$	Algeb	$-P_{sel} + \omega_g s_1^{SW_P} y_{Pfilt} + s_0^{SW_P} y_{Pfilt}$
VDL1_y	$y_{V_{DL1}}$	Algeb	$-y_{V_{DL1}} + \text{FixPiecewise}((I_{q1}, V_{q1} \geq y_{s0}), (I_{q1} + k_{Vq12} (-V_{q1} + y_{s0}), V_{q2} \geq y_{s0}), (I_{q1} + k_{Vq12} (-V_{q1} + y_{s0}), V_{q2} < y_{s0}), (I_{q1} + k_{Vq12} (-V_{q1} + y_{s0}), V_{q2} \geq y_{s0}), (I_{q1} + k_{Vq12} (-V_{q1} + y_{s0}), V_{q2} < y_{s0}))$
VDL2_y	$y_{V_{DL2}}$	Algeb	$-y_{V_{DL2}} + \text{FixPiecewise}((I_{p1}, V_{p1} \geq y_{s0}), (I_{p1} + k_{Vp12} (-V_{p1} + y_{s0}), V_{p2} \geq y_{s0}), (I_{p1} + k_{Vp12} (-V_{p1} + y_{s0}), V_{p2} < y_{s0}), (I_{p1} + k_{Vp12} (-V_{p1} + y_{s0}), V_{p2} \geq y_{s0}), (I_{p1} + k_{Vp12} (-V_{p1} + y_{s0}), V_{p2} < y_{s0}))$
Ipmax	$I_{pmax}$	Algeb	$-I_{pmax} + I_{pmax} h fThld_2 + (1 - fThld_2) \left( \sqrt{I_{pmax}^2 s_0^{SW_{PQ}}} + s_1^{SW_{PQ}} (z_{V_{DL2}} (I_{maxr} - I_{pmax}) + I_{pmax} (V_{p2} - V_{p1})) \right)$
Iqmax	$I_{qmax}$	Algeb	$\sqrt{I_{qmax}^2 s_1^{SW_{PQ}}} - I_{qmax} + s_0^{SW_{PQ}} (z_{V_{DL1}} (I_{maxr} (1 - V_{DL1c}) + V_{DL1c} y_{V_{DL1}}) - I_{qmax} (V_{q2} - V_{q1}))$
PIV_ys	$y_{sPIV}$	Algeb	$(1 - z_{Vdip}) \left( K_{vp} s_1^{SW_V} \left( -s_0^{SW_V} y_{s0} + y_{Vsel} \right) + xi_{PIV} - y_{sPIV} \right)$

Table 30 – continued from previous page

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
PIV_y	$y_{PIV}$	Algeb	$(1 - z_{Vdip}) \left( I_{qmax} z_u^{lim_{PIV}} + I_{qmin} z_l^{lim_{PIV}} - y_{PIV} + y_{SPIV} z_i^{lim_{PIV}} \right)$
Qsel	$Q_{sel}$	Algeb	$-Q_{sel} + s_0^{SW_V} y_{s4} + s_1^{SW_V} y_{PIV}$
IpHL_x	$x_{IpHL}$	Algeb	$-x_{IpHL} + \frac{y_{s5}}{V_p}$
IpHL_y	$y_{IpHL}$	Algeb	$I_{pmax} z_u^{lim_{IpHL}} + I_{pmin} z_l^{lim_{IpHL}} + x_{IpHL} z_i^{lim_{IpHL}} - y_{IpHL}$
IqHL_x	$x_{IqHL}$	Algeb	$I_{qinj} + Q_{sel} - x_{IqHL}$
IqHL_y	$y_{IqHL}$	Algeb	$I_{qmax} z_u^{lim_{IqHL}} + I_{qmin} z_l^{lim_{IqHL}} + x_{IqHL} z_i^{lim_{IqHL}} - y_{IqHL}$
a	$\theta$	ExtAlgeb	0
v	$V_d$	ExtAlgeb	0
Pe	$Pe$	ExtAlgeb	0
Qe	$Qe$	ExtAlgeb	0
Ipcmd	$Ipcmd$	ExtAlgeb	$-Ipcmd_0 + y_{IpHL}$
Iqcmd	$Iqcmd$	ExtAlgeb	$-Iqcmd_0 - y_{IqHL}$

## Services

Name	Symbol	Equation	Type
Ipcmd0	$Ipcmd0$	$\frac{P_0}{V_d}$	ConstService
Iqcmd0	$Iqcmd0$	$-\frac{Q_0}{V_d}$	ConstService
pfaref0	$\Phi_{ref0}$	$\text{atan}_2(Q_0, P_0)$	ConstService
Volt_dip	$z_{Vdip}$	$1 - z_i^{V_{cmp}}$	VarService
qref0	$q_{ref0}$	$Iqcmd_0 s_0^{SW_V} (V_d z_i^{VLower} + 0.01 z_l^{VLower}) + s_1^{SW_V} (V_d - V_{ref1})$	ConstService
PIQ_flag	$z_{PIQ}^{flag}$	0	EventFlag
s4_flag	$z_{s4}^{flag}$	0	EventFlag
pThld	$pThld$	Indicator( $T_{hld} > 0$ )	ConstService
nThld	$nThld$	Indicator( $T_{hld} < 0$ )	ConstService
Thld_abs	$ Thld $	$ T_{hld} $	ConstService
fThld	$fThld$	0	ExtendedEvent
s5_flag	$z_{s5}^{flag}$	0	EventFlag
kVq12	$k_{Vq12}$	$\frac{-I_{q1} + I_{q2}}{-V_{q1} + V_{q2}}$	ConstService
kVq23	$k_{Vq23}$	$\frac{-I_{q2} + I_{q3}}{-V_{q2} + V_{q3}}$	ConstService
kVq34	$k_{Vq34}$	$\frac{-I_{q3} + I_{q4}}{-V_{q3} + V_{q4}}$	ConstService
zVDL1	$z_{VDL1}$	$I_{q1} \leq I_{q2} \wedge I_{q2} \leq I_{q3} \wedge I_{q3} \leq I_{q4} \wedge V_{q1} \leq V_{q2} \wedge V_{q2} \leq V_{q3} \wedge V_{q3} \leq V_{q4}$	ConstService
kVp12	$k_{Vp12}$	$\frac{-I_{p1} + I_{p2}}{-V_{p1} + V_{p2}}$	ConstService
kVp23	$k_{Vp23}$	$\frac{-I_{p2} + I_{p3}}{-V_{p2} + V_{p3}}$	ConstService
kVp34	$k_{Vp34}$	$\frac{-I_{p3} + I_{p4}}{-V_{p3} + V_{p4}}$	ConstService
zVDL2	$z_{VDL2}$	$I_{p1} \leq I_{p2} \wedge I_{p2} \leq I_{p3} \wedge I_{p3} \leq I_{p4} \wedge V_{p1} \leq V_{p2} \wedge V_{p2} \leq V_{p3} \wedge V_{p3} \leq V_{p4}$	ConstService
fThld2	$fThld2$	0	ExtendedEvent
VDL1c	$VDL1c$	$y_{V_{DL1}} < I_{maxr}$	VarService

continues on next page

Table 31 – continued from previous page

Name	Symbol	Equation	Type
VDL2c	$VDL2c$	$y_{VDL2} < I_{maxr}$	VarService
Ipmax2sq0	$I_{pmax20,nn}^2$	FixPiecewise $((0, I_{max}^2 - I_{qcmd0}^2 \leq 0), (I_{max}^2 - I_{qcmd0}^2, \text{True}))$	ConstService
Ipmax2sq	$I_{pmax2}^2$	FixPiecewise $((0, I_{max}^2 - y_{IqHL}^2 \leq 0), (I_{max}^2 - y_{IqHL}^2, \text{True}))$	VarService
Ipmaxh	$Ipmaxh$	0	VarHold
Iqmax2sq0	$I_{qmax,nn}^2$	FixPiecewise $((0, I_{max}^2 - I_{pcmd0}^2 \leq 0), (I_{max}^2 - I_{pcmd0}^2, \text{True}))$	ConstService
Iqmax2sq	$I_{qmax2}^2$	FixPiecewise $((0, I_{max}^2 - y_{IpHL}^2 \leq 0), (I_{max}^2 - y_{IpHL}^2, \text{True}))$	VarService
Ipmin	$I_{pmin}$	0.0	ConstService
PIV_flag	$z_{PIV}^{flag}$	0	EventFlag

## 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	$lim_{Pfilt}$	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	$s_0$	Lag	Voltage filter
S1	$S_1$	Lag	Pe filter
PIQ	$PIQ$	PITrackAWFreeze	
Vsel	$V_{sel}$	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	$s_5$	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	$I_{pHL}$	GainLimiter	
IqHL	$I_{qHL}$	GainLimiter	

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	

## 5.24 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  $S_n$ . When replacing a synchronous generator, the renewable generator should have the same or larger  $S_n$ . Improper  $S_n$  will cause the initial values to exceed typical per-unit ranges and cause the initialization to fail.

Common Parameters: u, name, bus, gen,  $S_n$

Common Variables: Pe, Qe

Available models: [REGCA1](#), [REGCP1](#), [REGCV1](#), [REGCV2](#), [REGF1](#), [REGF2](#), [REGF3](#)

### 5.24.1 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 ( $\text{Volim} > v$ ), 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	$Z_{zerox}$	LVPL characteristic voltage 1	0.500	p.u	
Lv-plsw	$z_{Lvplesw}$	Low volt. P logic: 1-enable, 0-disable	1	bool	
Lvp11	$L_{vp11}$	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. (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		
Iqr-max	$I_{qrmax}$	Upper limit on the ROC for reactive current	1	p.u.	current
Iqr-min	$I_{qrmin}$	Lower limit on the ROC for reactive current	-1	p.u.	current
Accel	$A_{accel}$	Acceleration factor	0		
gammaP	$\gamma_P$	P ratio of linked static gen	1		
gammaQ	$\gamma_Q$	Q ratio of linked static gen	1		
ra	$r_a$		0		
xs	$x_s$		0		

## Variables

Name	Symbol	Type	Description	Unit	Properties
S1_y	$y_{S_1}$	State	State in lag TF	v_str	
S2_y	$y_{S_2}$	State	State in lag transfer function	v_str	
S0_y	$y_{S_0}$	State	State in lag TF	v_str	
LVG_y	$y_{LVG}$	Algeb	Output of piecewise	v_str	
Ipcmd	$I_{pcmd}$	Algeb	current component for active power	v_str	
Iqcmd	$I_{qcmd}$	Algeb	current component for reactive power	v_str	
LVPL_y	$y_{LVPL}$	Algeb	Output of piecewise	v_str	
Ipout	$I_{pout}$	Algeb	Output Ip current	v_str	
HVG_x	$x_{HVG}$	Algeb	Value before limiter	v_str	
HVG_y	$y_{HVG}$	Algeb	Output after limiter and post gain	v_str	
Iqout_x	$x_{Iqout}$	Algeb	Value before limiter	v_str	
Iqout_y	$y_{Iqout}$	Algeb	Output after limiter and post gain	v_str	
Pe	$P_e$	Algeb	Active power output	v_str	
Qe	$Q_e$	Algeb	Reactive power output	v_str	
a	$\theta$	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	$y_{S_1}$	State	$-I_{qcmd}$
S2_y	$y_{S_2}$	State	$1.0V$
S0_y	$y_{S_0}$	State	$I_{pcmd}$
LVG_`	$y_{LVG}$	Algeb	FixPiecewise ((0, $L_{vpnt0} \geq V$ ), ( $k_{LVG} (-L_{vpnt0} + V)$ , $L_{vpnt1} \geq V$ ), (1, True))
Ipcmd	$I_{pcmd}$	Algeb	$\frac{I_{pcmd0} \text{Indicator}(y_{LVG} > 0)}{y_{LVG}} + \text{Indicator}(y_{LVG} \leq 0)$
Iqcmd	$I_{qcmd}$	Algeb	$I_{qcmd0}$
LVPL	$y_{LVPL}$	Algeb	FixPiecewise ((9999 - 9999 $z_{Lvppls_w}$ , $Z_{erox} \geq y_{S_2}$ ), ( $k_{LVPL} (-Z_{erox} + y_{S_2}) - 9999z_{Lvppls_w} + 9999$ , $Z_{erox} < y_{S_2}$ ))
Ipout	$I_{pout}$	Algeb	$I_{pcmdy_{LVG}}$
HVG_	$x_{HVG}$	Algeb	$K_{hv} (V - V_{olim})$
HVG_	$y_{HVG}$	Algeb	$x_{HVG} z_i^{\lim_{HVG}}$
Iqout_	$x_{Iqout}$	Algeb	$-y_{HVG} + y_{S_1}$
Iqout_	$y_{Iqout}$	Algeb	$I_{olim} z_l^{\lim_{Iqout}} + x_{Iqout} z_i^{\lim_{Iqout}}$
Pe	$P_e$	Algeb	$P_0$
Qe	$Q_e$	Algeb	$Q_0$
a	$\theta$	Ex-Algeb	
v	$V$	Ex-Algeb	
vd	$vd$	AliasAlgeb	

## Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
S1_y	$y_{S_1}$	State	$-I_{qcmd} - y_{S_1}$	$T_g$
S2_y	$y_{S_2}$	State	$1.0V - y_{S_2}$	$T_{fltr}$
S0_y	$y_{S_0}$	State	$I_{pcmd} - y_{S_0}$	$T_g$

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
LVG_	$y_{LVG}$	Algeb	$-y_{LVG} + \text{FixPiecewise}((0, L_{vpnt0} \geq V), (k_{LVG}(-L_{vpnt0} + V), L_{vpnt1} \geq V), (1, \text{True}))$
Ipcmd	$I_{pcmd}$	Algeb	$-I_{pcmd} + I_{pcmd0}_{LVG}$
Iqcmd	$I_{qcmd}$	Algeb	$I_{qcmd0} - I_{qcmd}$
LVPL	$y_{LVPL}$	Algeb	$-y_{LVPL} + \text{FixPiecewise}((9999 - 9999z_{Lvplesw}, Z_{erex} \geq y_{S2}), (k_{LVPL}(-Z_{erex} + y_{S2}) - 9999z_{Lvplesw}, Z_{erex} < y_{S2})), (0, \text{True}))$
Ipout	$I_{pout}$	Algeb	$-I_{pout} + y_{LVG}y_{S0}$
HVG_	$x_{HVG}$	Algeb	$K_{hv}(V - V_{olim}) - x_{HVG}$
HVG_	$y_{HVG}$	Algeb	$x_{HVG}z_i^{lim_{HVG}} - y_{HVG}$
Iqout_	$x_{Iqout}$	Algeb	$-x_{Iqout} - y_{HVG} + y_{S1}$
Iqout_	$y_{Iqout}$	Algeb	$I_{olim}z_l^{lim_{Iqout}} + x_{Iqout}z_i^{lim_{Iqout}} - y_{Iqout}$
Pe	$P_e$	Algeb	$I_{pout}V - P_e$
Qe	$Q_e$	Algeb	$-Q_e + Vy_{Iqout}$
a	$\theta$	ExtAlgebra	$-P_e$
v	$V$	ExtAlgebra	$-Q_e$
vd	$vd$	AliasAlgebra	0

## Services

Name	Symbol	Equation	Type
p0	$P_0$	$P_{0s}\gamma_P$	ConstService
q0	$Q_0$	$Q_{0s}\gamma_Q$	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	$I_{pcmd0}_{LVG}$	$I_{pcmd}$	PostInitService
kLVG	$k_{LVG}$	$\frac{1}{-L_{vpnt0} + L_{vpnt1}}$	ConstService
kLVPL	$k_{LVPL}$	$\frac{L_{vpnt1}z_{Lvplesw}}{B_{rkpt} - Z_{erex}}$	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	$lim_{HVG}$	HardLimiter	
Iqout_lim	$lim_{Iqout}$	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	$LVPL$	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)

### 5.24.2 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	De-fault	Unit	Proper-ties
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	$Z_{zerox}$	LVPL characteristic voltage 1	0.500	p.u	
Lv-plsw	$z_{Lvplesw}$	Low volt. P logic: 1-enable, 0-disable	1	bool	
Lvp11	$L_{vp11}$	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		
Iqr-max	$I_{qrmax}$	Upper limit on the ROC for reactive current	1	p.u.	current
Iqr-min	$I_{qrmin}$	Lower limit on the ROC for reactive current	-1	p.u.	current
Accel	$A_{accel}$	Acceleration factor	0		
gammaP	$\gamma_P$	P ratio of linked static gen	1		
gammaQ	$\gamma_Q$	Q ratio of linked static gen	1		
pll		Phase-lock loop device idx			
ra	$r_a$		0		
xs	$x_s$		0		

## Variables

Name	Symbol	Type	Description	Unit	Properties
S1_y	$y_{S_1}$	State	State in lag TF	v_str	
S2_y	$y_{S_2}$	State	State in lag transfer function	v_str	
S0_y	$y_{S_0}$	State	State in lag TF	v_str	
am	$\theta_m$	ExtState	Measured angle		
LVG_y	$y_{LVG}$	Algeb	Output of piecewise	v_str	
Ipcmd	$I_{pcmd}$	Algeb	current component for active power	v_str	
Iqcmd	$I_{qcmd}$	Algeb	current component for reactive power	v_str	
LVPL_y	$y_{LVPL}$	Algeb	Output of piecewise	v_str	
Ipout	$I_{pout}$	Algeb	Output Ip current	v_str	
HVG_x	$x_{HVG}$	Algeb	Value before limiter	v_str	
HVG_y	$y_{HVG}$	Algeb	Output after limiter and post gain	v_str	
Iqout_x	$x_{Iqout}$	Algeb	Value before limiter	v_str	
Iqout_y	$y_{Iqout}$	Algeb	Output after limiter and post gain	v_str	
Pe	$P_e$	Algeb	Active power output	v_str	
Qe	$Q_e$	Algeb	Reactive power output	v_str	
vd	$V_d$	Algeb	d-axis voltage	v_str	
vq	$V_q$	Algeb	q-axis voltage	v_str	
a	$\theta$	ExtAlgeb	Bus voltage angle		
v	$V$	ExtAlgeb	Bus voltage magnitude		

## Initialization Equations

Name	Sym- bol	Type	Initial Value
S1_y	$y_{S_1}$	State	$-I_{qcmd}$
S2_y	$y_{S_2}$	State	$1.0V$
S0_y	$y_{S_0}$	State	$I_{pcmd}$
am	$\theta_m$	ExtStat	
LVG_-	$y_{LVG}$	Al- geb	FixPiecewise ((0, $L_{vpnt0} \geq V$ ), ( $k_{LVG} (-L_{vpnt0} + V)$ , $L_{vpnt1} \geq V$ ), (1, True))
Ipcmd	$I_{pcmd}$	Al- geb	$\frac{I_{pcmd0} \text{Indicator}(y_{LVG} > 0)}{y_{LVG}} + \text{Indicator}(y_{LVG} \leq 0)$
Iqcmd	$I_{qcmd}$	Al- geb	$I_{qcmd0}$
LVPL_-	$y_{LVPL}$	Al- geb	FixPiecewise ((9999 - 9999 $z_{Lvplsw}$ , $Z_{erox} \geq y_{S_2}$ ), ( $k_{LVPL} (-Z_{erox} + y_{S_2}) - 9999z_{Lvplsw} + 9999$ , $Z_{erox} < y_{S_2}$ ))
Ipout	$I_{pout}$	Al- geb	$I_{pcmd} y_{LVG}$
HVG_-	$x_{HVG}$	Al- geb	$K_{hv} (V - V_{olim})$
HVG_-	$y_{HVG}$	Al- geb	$x_{HVG} z_i^{\lim_{HVG}}$
Iqout_-	$x_{Iqout}$	Al- geb	$-y_{HVG} + y_{S_1}$
Iqout_-	$y_{Iqout}$	Al- geb	$I_{olim} z_l^{\lim_{Iqout}} + x_{Iqout} z_i^{\lim_{Iqout}}$
Pe	$P_e$	Al- geb	$P_0$
Qe	$Q_e$	Al- geb	$Q_0$
vd	$V_d$	Al- geb	$V$
vq	$V_q$	Al- geb	0
a	$\theta$	Ex- tAl- geb	
v	$V$	Ex- tAl- geb	

## Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
S1_y	$y_{S_1}$	State	$-I_{qcmd} - y_{S_1}$	$T_g$
S2_y	$y_{S_2}$	State	$1.0V - y_{S_2}$	$T_{fltr}$
S0_y	$y_{S_0}$	State	$I_{pcmd} - y_{S_0}$	$T_g$
am	$\theta_m$	ExtState	0	

## Algebraic Equations

Name	Sym- bol	Type	RHS of Equation "0 = g(x, y)"
LVG_-	$y_{LVG}$	Al- geb bol	$-y_{LVG} + \text{FixPiecewise}\left((0, L_{vpnt0} \geq V), (k_{LVG}(-L_{vpnt0} + V), L_{vpnt1} \geq V), (1, \text{True})\right)$
Ipcmd	$I_{pcmd}$	Al- geb bol	$-I_{pcmd} + I_{pcmd0}_{LVG}$
Iqcmd	$I_{qcmd}$	Al- geb bol	$I_{qcmd0} - I_{qcmd}$
LVPL	$y_{LVPL}$	Al- geb bol	$-y_{LVPL} + \text{FixPiecewise}\left((9999 - 9999z_{Lvppls}, Z_{er0x} \geq y_{S2}), (k_{LVPL}(-Z_{er0x} + y_{S2}) - 9999z_{Lvppls}, Z_{er0x} < y_{S2})\right)$
Ipout	$I_{pout}$	Al- geb bol	$-I_{pout} + y_{LVG}y_{S0}$
HVG_-	$x_{HVG}$	Al- geb bol	$K_{hv}(V - V_{olim}) - x_{HVG}$
HVG_-	$y_{HVG}$	Al- geb bol	$x_{HVG} z_i^{\lim_{HVG}} - y_{HVG}$
Iqout_-	$x_{Iqout}$	Al- geb bol	$-x_{Iqout} - y_{HVG} + y_{S1}$
Iqout_-	$y_{Iqout}$	Al- geb bol	$I_{olim} z_l^{\lim_{Iqout}} + x_{Iqout} z_i^{\lim_{Iqout}} - y_{Iqout}$
Pe	$P_e$	Al- geb bol	$I_{pout}V_d - P_e + V_q y_{Iqout}$
Qe	$Q_e$	Al- geb bol	$-I_{pout}V_q - Q_e + V_d y_{Iqout}$
vd	$V_d$	Al- geb bol	$V \cos(z_{hetam}(\theta - \theta_m)) - V_d$
vq	$V_q$	Al- geb bol	$-V \sin(z_{hetam}(\theta - \theta_m)) - V_q$
a	$\theta$	Ex- tAl- geb bol	$-P_e$
v	$V$	Ex- tAl- geb bol	$-Q_e$

## Services

Name	Symbol	Equation	Type
p0	$P_0$	$P_{0s} \gamma_P$	ConstService
q0	$Q_0$	$Q_{0s} \gamma_Q$	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	$I_{pcmd0LVG}$	$I_{pcmd}$	PostInitService
kLVG	$k_{LVG}$	$\frac{1}{-L_{vpnt0} + L_{vpnt1}}$	ConstService
kLVPL	$k_{LVPL}$	$\frac{L_{vpl1} z_{Lvpns}}{B_{rkpt} - Z_{erox}}$	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	$lim_{HVG}$	HardLimiter	
Iqout_lim	$lim_{Iqout}$	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)

### 5.24.3 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  $k_w$  can improve small-signal stability.
- The droop  $k_v$  for voltage control (pu voltage / pu Q change), if used, needs to be chosen carefully. In most cases,  $k_v$  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	$k_\omega$	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	$\gamma_P$	P ratio of linked static gen	1		
gam-	$\gamma_Q$	Q ratio of linked static gen	1		
maq					
Kpvd	$k_{p_{vd}}$	vd controller proportional gain	0.500	p.u.	power
Kivd	$k_{i_{vd}}$	vd controller integral gain	0.020	p.u.	power
Kpvq	$k_{p_{vq}}$	vq controller proportional gain	0.500	p.u.	power
Kivq	$k_{i_{vq}}$	vq controller integral gain	0.020	p.u.	power
KpId	$k_{p_{di}}$	Id controller proportional gain	0.200	p.u.	power
KiId	$k_{i_{di}}$	Id controller integral gain	0.010	p.u.	power
KpIq	$k_{p_{qi}}$	Iq controller proportional gain	0.200	p.u.	power
KiIq	$k_{i_{qi}}$	Iq controller integral gain	0.010	p.u.	power

## Variables

Name	Symbol	Type	Description	Unit	Properties
dw	$\Delta\omega$	State	delta virtual rotor speed	$pu$ (Hz)	v_str
delta	$\delta$	State	virtual delta	rad	v_str
PIvd_xi	$xi_{PIvd}$	State	Integrator output		v_str
PIvq_xi	$xi_{PIvq}$	State	Integrator output		v_str
PIId_xi	$xi_{PIId}$	State	Integrator output		v_str
PIIq_xi	$xi_{PIIq}$	State	Integrator output		v_str
ud-	$y_{udLag}$	State	State in lag transfer function		v_str
Lag_y					
uqLag_y	$y_{uqLag}$	State	State in lag transfer function		v_str
ud	$ud$	AliasState	Alias of udLag_y		
uq	$uq$	AliasState	Alias of uqLag_y		
Pref2	$P_{ref2}$	Algeb	active power reference after adjusting by frequency		v_str
vref2	$v_{ref2}$	Algeb	voltage reference after adjusted by reactive power		v_str
omega	$\omega$	Algeb	virtual rotor speed	$pu$ (Hz)	v_str
vd	$V_d$	Algeb	d-axis voltage		v_str
vq	$V_q$	Algeb	q-axis voltage		v_str
Pe	$P_e$	Algeb	active power injection from VSC		v_str
Qe	$Q_e$	Algeb	reactive power injection from VSC		v_str
Id	$I_d$	Algeb	d-axis current		v_str
Iq	$I_q$	Algeb	q-axis current		v_str
PIvd_y	$y_{PIvd}$	Algeb	PI output		v_str
PIvq_y	$y_{PIvq}$	Algeb	PI output		v_str
PIId_y	$y_{PIId}$	Algeb	PI output		v_str
PIIq_y	$y_{PIIq}$	Algeb	PI output		v_str
udref	$u_{dref}$	Algeb	ud reference		v_str
uqref	$u_{qref}$	Algeb	uq reference		v_str
a	$\theta$	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	$\Delta\omega$	State	0
delta	$\delta$	State	$\theta$
PIvd_xi	$x_{PIvd}$	State	$I_{d0}$
PIvq_xi	$x_{PIvq}$	State	$I_{q0}$
PIId_xi	$x_{PIId}$	State	0.0
PIIq_xi	$x_{PIIq}$	State	0.0
udLag_y	$y_{udLag}$	State	$udref$
uqLag_y	$y_{uqLag}$	State	$uqref$
ud	$ud$	AliasState	
uq	$uq$	AliasState	
Pref2	$P_{ref2}$	Algeb	$P_{ref}u$
vref2	$v_{ref2}$	Algeb	$V_{ref}u$
omega	$\omega$	Algeb	$u$
vd	$V_d$	Algeb	$v_{d0}$
vq	$V_q$	Algeb	$v_{q0}$
Pe	$P_e$	Algeb	$P_{ref}$
Qe	$Q_e$	Algeb	$Q_{ref}$
Id	$I_d$	Algeb	$I_{d0}$
Iq	$I_q$	Algeb	$I_{q0}$
PIvd_y	$y_{PIvd}$	Algeb	$I_{d0} + kp_{vd}(V_d - v_{ref2})$
PIvq_y	$y_{PIvq}$	Algeb	$I_{q0} + V_q kp_{vq}$
PIId_y	$y_{PIId}$	Algeb	$kp_{di}(I_d - y_{PIvd})$
PIIq_y	$y_{PIIq}$	Algeb	$kp_{qi}(I_q - y_{PIvq})$
udref	$udref$	Algeb	$udref0$
uqref	$uqref$	Algeb	$uqref0$
a	$\theta$	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	$\Delta\omega$	State	$-D\Delta\omega - P_e + P_{ref2}$	$M$
delta	$\delta$	State	$2\pi\Delta\omega f$	
PIvd_xi	$xi_{PIvd}$	State	$ki_{vd}(V_d - v_{ref2})$	
PIvq_xi	$xi_{PIvq}$	State	$V_q ki_{vq}$	
PIId_xi	$xi_{PIId}$	State	$ki_{di}(I_d - y_{PIvd})$	
PIIq_xi	$xi_{PIIq}$	State	$ki_{qi}(I_q - y_{PIvq})$	
udLag_y	$y_{udLag}$	State	$u_{dref} - y_{udLag}$	$T_c$
uqLag_y	$y_{uqLag}$	State	$u_{qref} - y_{uqLag}$	$T_c$
ud	$ud$	AliasState	0	
uq	$uq$	AliasState	0	

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
Pref2	$P_{ref2}$	Algeb	$-P_{ref2} + P_{ref}u - \Delta\omega k_\omega$
vref2	$v_{ref2}$	Algeb	$V_{ref} + k_v(-Q_e + Q_{ref}u) - v_{ref2}$
omega	$\omega$	Algeb	$\Delta\omega - \omega + 1$
vd	$V_d$	Algeb	$V_u \cos(\delta - \theta) - V_d$
vq	$V_q$	Algeb	$-V_u \sin(\delta - \theta) - V_q$
Pe	$P_e$	Algeb	$I_d V_d + I_q V_q - P_e$
Qe	$Q_e$	Algeb	$I_d V_q - I_q V_d - Q_e$
Id	$I_d$	Algeb	$I_d r_a - I_q x_s + V_d - y_{udLag}$
Iq	$I_q$	Algeb	$I_d x_s + I_q r_a + V_q - y_{uqLag}$
PIvd_y	$y_{PIvd}$	Algeb	$k p_{vd}(V_d - v_{ref2}) + x_{PIvd} - y_{PIvd}$
PIvq_y	$y_{PIvq}$	Algeb	$V_q k p_{vq} + x_{PIvq} - y_{PIvq}$
PIId_y	$y_{PIId}$	Algeb	$k p_{di}(I_d - y_{PIvd}) + x_{PIId} - y_{PIId}$
PIIq_y	$y_{PIIq}$	Algeb	$k p_{qi}(I_q - y_{PIvq}) + x_{PIIq} - y_{PIIq}$
udref	$u_{dref}$	Algeb	$-I_{qref} x_s + V_d - u_{dref} + y_{PIId}$
uqref	$u_{qref}$	Algeb	$I_{dref} x_s + V_q - u_{qref} + y_{PIIq}$
a	$\theta$	ExtAlgeb	$-P_e u$
v	$V$	ExtAlgeb	$-Q_e u$
Idref	$Idref$	AliasAlgeb	0
Iqref	$Iqref$	AliasAlgeb	0

## Services

Name	Symbol	Equation	Type
Pref	$P_{ref}$	$P_{0s}\gamma_P$	ConstService
Qref	$Q_{ref}$	$Q_{0s}\gamma_Q$	ConstService
ixs	$1/x_s$	$\frac{1}{x_s}$	ConstService
Id0	$I_{d0}$	$\frac{P_{ref}u}{V}$	ConstService
Iq0	$I_{q0}$	$-\frac{Q_{ref}u}{V}$	ConstService
vd0	$v_{d0}$	$Vu$	ConstService
vq0	$v_{q0}$	0	ConstService
udref0	$u_{dref0}$	$I_{d0}r_a - I_{q0}x_s + v_{d0}$	ConstService
uqref0	$u_{qref0}$	$I_{d0}x_s + I_{q0}r_a + v_{q0}$	ConstService

## Blocks

Name	Symbol	Type	Info
PIvd	$PIvd$	PIController	
PIvq	$PIvq$	PIController	
PIId	$PIId$	PIController	
PIIq	$PIIq$	PIController	
udLag	$udLag$	Lag	
uqLag	$uqLag$	Lag	

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)

### 5.24.4 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  $K_{pvd}$ ,  $K_{ivd}$ ,  $K_{pvq}$ , and  $K_{ivq}$ , 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	$k_\omega$	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	$\gamma_P$	P ratio of linked static gen	1		
gam-	$\gamma_Q$	Q ratio of linked static gen	1		
maq					
Kpvd	$k_{pvd}$	vd controller proportional gain	0.500	p.u.	power
Kivd	$k_{ivd}$	vd controller integral gain	0.020	p.u.	power
Kpvq	$k_{pvq}$	vq controller proportional gain	0.500	p.u.	power
Kivq	$k_{ivq}$	vq controller integral gain	0.020	p.u.	power
Tiq	$T_{Iq}$		0.010		
Tid	$T_{Id}$		0.010		

## Variables

Name	Symbol	Type	Description	Unit	Properties
dw	$\Delta\omega$	State	delta virtual rotor speed	$pu$ (Hz)	v_str
delta	$\delta$	State	virtual delta	rad	v_str
PIvd_xi	$xi_{PIvd}$	State	Integrator output		v_str
PIvq_xi	$xi_{PIvq}$	State	Integrator output		v_str
LGId_y	$y_{LGId}$	State	State in lag transfer function		v_str
LGIq_y	$y_{LGIq}$	State	State in lag transfer function		v_str
Pref2	$P_{ref2}$	Algeb	active power reference after adjusting by frequency		v_str
vref2	$v_{ref2}$	Algeb	voltage reference after adjusted by reactive power		v_str
omega	$\omega$	Algeb	virtual rotor speed	$pu$ (Hz)	v_str
vd	$V_d$	Algeb	d-axis voltage		v_str
vq	$V_q$	Algeb	q-axis voltage		v_str
Pe	$P_e$	Algeb	active power injection from VSC		v_str
Qe	$Q_e$	Algeb	reactive power injection from VSC		v_str
Id	$I_d$	Algeb	d-axis current		v_str
Iq	$I_q$	Algeb	q-axis current		v_str
PIvd_y	$y_{PIvd}$	Algeb	PI output		v_str
PIvq_y	$y_{PIvq}$	Algeb	PI output		v_str
a	$\theta$	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	$\Delta\omega$	State	0
delta	$\delta$	State	$\theta$
PIvd_xi	$xi_{PIvd}$	State	$I_{d0}$
PIvq_xi	$xi_{PIvq}$	State	$I_{q0}$
LGId_y	$y_{LGId}$	State	$y_{PIvd}$
LGIq_y	$y_{LGIq}$	State	$y_{PIvq}$
Pref2	$P_{ref2}$	Algeb	$P_{refu}$
vref2	$v_{ref2}$	Algeb	$V_{refu}$
omega	$\omega$	Algeb	$u$
vd	$V_d$	Algeb	$v_{d0}$
vq	$V_q$	Algeb	$v_{q0}$
Pe	$P_e$	Algeb	$P_{ref}$
Qe	$Q_e$	Algeb	$Q_{ref}$
Id	$I_d$	Algeb	$I_{d0}$
Iq	$I_q$	Algeb	$I_{q0}$
PIvd_y	$y_{PIvd}$	Algeb	$I_{d0} + kp_{vd}(V_d - v_{ref2})$
PIvq_y	$y_{PIvq}$	Algeb	$I_{q0} + V_q kp_{vq}$
a	$\theta$	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	$\Delta\omega$	State	$-D\Delta\omega - P_e + P_{ref2}$	$M$
delta	$\delta$	State	$2\pi\Delta\omega f$	
PIvd_xi	$xi_{PIvd}$	State	$ki_{vd}(V_d - v_{ref2})$	
PIvq_xi	$xi_{PIvq}$	State	$V_q ki_{vq}$	
LGId_y	$y_{LGId}$	State	$-y_{LGId} + y_{PIvd}$	$T_{Id}$
LGIq_y	$y_{LGIq}$	State	$-y_{LGIq} + y_{PIvq}$	$T_{Iq}$

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
Pref2	$P_{ref2}$	Algeb	$-P_{ref2} + P_{refu} - \Delta\omega k_\omega$
vref2	$v_{ref2}$	Algeb	$V_{ref} + k_v (-Q_e + Q_{refu}) - v_{ref2}$
omega	$\omega$	Algeb	$\Delta\omega - \omega + 1$
vd	$V_d$	Algeb	$V_u \cos(\delta - \theta) - V_d$
vq	$V_q$	Algeb	$-V_u \sin(\delta - \theta) - V_q$
Pe	$P_e$	Algeb	$I_d V_d + I_q V_q - P_e$
Qe	$Q_e$	Algeb	$I_d V_q - I_q V_d - Q_e$
Id	$I_d$	Algeb	$-I_d + yLGI_d$
Iq	$I_q$	Algeb	$-I_q + yLGI_q$
PIvd_y	$yPIvd$	Algeb	$k p_{vd} (V_d - v_{ref2}) + x i_{PIvd} - y_{PIvd}$
PIvq_y	$yPIvq$	Algeb	$V_q k p_{vq} + x i_{PIvq} - y_{PIvq}$
a	$\theta$	ExtAlgeb	$-P_e u$
v	$V$	ExtAlgeb	$-Q_e u$
Idref	$Idref$	AliasAlgeb	0
Iqref	$Iqref$	AliasAlgeb	0

## Services

Name	Symbol	Equation	Type
Pref	$P_{ref}$	$P_{0s}\gamma_P$	ConstService
Qref	$Q_{ref}$	$Q_{0s}\gamma_Q$	ConstService
ixs	$1/xs$	$\frac{1}{x_s}$	ConstService
Id0	$I_{d0}$	$\frac{P_{refu}}{V}$	ConstService
Iq0	$I_{q0}$	$-\frac{Q_{refu}}{V}$	ConstService
vd0	$v_{d0}$	$V_u$	ConstService
vq0	$v_{q0}$	0	ConstService

## Blocks

Name	Symbol	Type	Info
PIvd	$PIvd$	PIController	
PIvq	$PIvq$	PIController	
LGI <sub>d</sub>	$LGI_d$	Lag	
LGI <sub>q</sub>	$LGI_q$	Lag	

Config Fields in [RECGV2]

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)

## 5.24.5 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	<i>bool</i>	
name		device name			
bus		interface bus id			mandatory
gen		static generator index			mandatory
Sn	$S_n$	Model MVA base	100	<i>MVA</i>	
rf	$r_a$	resistance	0	<i>z</i>	
xf	$x_s$	reactance	0.200	<i>z</i>	
Vdip	$V_{dip}$	V threshold to freeze states	0.800	<i>p.u.</i>	
Tfrz	$T_{frz}$	Time to keep state frozen	0		
PQFLAG		P/Q priority flag; 0-Q priority, 1-P priority	1	<i>bool</i>	
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	$\omega_{drp}$	frequency droop percentage	0.033		
Qdrp	$Q_{drp}$	Voltage droop percentage	0.045		
Tr	$T_c$	transducer time constant	0.005	<i>s</i>	
Te	$T_e$	output state time constant	0.005	<i>s</i>	
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

continues on next page

Table 32 – continued from previous page

Name	Symbol	Description	Default	Unit	Properties
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	$\gamma_P$	P ratio of linked static gen	1		
gammaq	$\gamma_Q$	Q ratio of linked static gen	1		

## Variables

Name	Symbol	Type	Description	Unit	Properties
Psen_y	$y_{Psen}$	State	State in lag transfer function	v_str	
Qsen_y	$y_{Qsen}$	State	State in lag transfer function	v_str	
Psig_y	$y_{Psig}$	State	State in lag TF	v_str	
Qsig_y	$y_{Qsig}$	State	State in lag TF	v_str	
PIplim_xi	$x_{iPIplim}$	State	Integrator output	v_str	
PIqlim_xi	$x_{iPIqlim}$	State	Integrator output	v_str	
delta	$\delta$	State	virtual delta	rad	v_str
PIvd_xi	$x_{iPIvd}$	State	Integrator output	v_str	
PIvq_xi	$x_{iPIvq}$	State	Integrator output	v_str	
PIId_xi	$x_{iPIId}$	State	Integrator output	v_str	
PIIq_xi	$x_{iPIIq}$	State	Integrator output	v_str	
udLag_y	$y_{udLag}$	State	State in lag transfer function	v_str	
uqLag_y	$y_{uqLag}$	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	$y_{PIplim}$	Algeb	PI output	v_str	
PIqlim_y	$y_{PIqlim}$	Algeb	PI output	v_str	
vd	$V_d$	Algeb	d-axis voltage	v_str	
vq	$V_q$	Algeb	q-axis voltage	v_str	
Pe	$P_e$	Algeb	active power injection from VSC	v_str	
Qe	$Q_e$	Algeb	reactive power injection from VSC	v_str	
Id	$I_d$	Algeb	d-axis current	v_str	
Iq	$I_q$	Algeb	q-axis current	v_str	
dw_x	$x_{dw}$	Algeb	Value before limiter	v_str	
dw_y	$y_{dw}$	Algeb	Output after limiter and post gain	v_str	
vref2	$v_{ref2}$	Algeb	voltage reference after droop	v_str	
PIvd_y	$y_{PIvd}$	Algeb	PI output	v_str	
PIvq_y	$y_{PIvq}$	Algeb	PI output	v_str	

continues on next page

Table 33 – continued from previous page

Name	Symbol	Type	Description	Unit	Properties
PIId_y	$y_{PIId}$	Algeb	PI output		v_str
PIIq_y	$y_{PIIq}$	Algeb	PI output		v_str
udref	$u_{dref}$	Algeb	ud reference		v_str
uqref	$u_{qref}$	Algeb	uq reference		v_str
a	$\theta$	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	$y_{Psen}$	State	$P_e$
Qsen_y	$y_{Qsen}$	State	$Q_e$
Psig_y	$y_{Psig}$	State	$P_{aux} + y_{Psen}$
Qsig_y	$y_{Qsig}$	State	$Q_{aux} + y_{Qsen}$
PIplim_xi	$xi_{PIplim}$	State	$y_{Psen}$
PIqlim_xi	$xi_{PIqlim}$	State	$y_{Qsen}$
delta	$\delta$	State	$\theta$
PIvd_xi	$xi_{PIvd}$	State	$I_{d0}$
PIvq_xi	$xi_{PIvq}$	State	$I_{q0}$
PIId_xi	$xi_{PIId}$	State	0.0
PIIq_xi	$xi_{PIIq}$	State	0.0
udLag_y	$y_{udLag}$	State	$u_{dref}$
uqLag_y	$y_{uqLag}$	State	$u_{qref}$
ud	$ud$	AliasState	
uq	$uq$	AliasState	
Paux	$P_{aux}$	Algeb	0
Qaux	$Q_{aux}$	Algeb	0
PIplim_y	$y_{PIplim}$	Algeb	$K_{Pplim}(-y_{Psen} + y_{Psig}) + y_{Psen}$
PIqlim_y	$y_{PIqlim}$	Algeb	$K_{Pqlim}(-y_{Qsen} + y_{Qsig}) + y_{Qsen}$
vd	$V_d$	Algeb	$v_{d0}$
vq	$V_q$	Algeb	$v_{q0}$
Pe	$P_e$	Algeb	$P_{ref}$
Qe	$Q_e$	Algeb	$Q_{ref}$
Id	$I_d$	Algeb	$I_{d0}$
Iq	$I_q$	Algeb	$I_{q0}$
dw_x	$x_{dw}$	Algeb	$\omega_{drp}w_0(y_{PIplim} - y_{Psen})$
dw_y	$y_{dw}$	Algeb	$\Delta\omega_{max}z_u^{lim_{dw}} + \Delta\omega_{min}z_l^{lim_{dw}} + x_{dw}z_i^{lim_{dw}}$
vref2	$v_{ref2}$	Algeb	$V_{ref}u$
PIvd_y	$y_{PIvd}$	Algeb	$I_{d0} + K_Pv(-V_d + v_{ref2})$

continues on next page

Table 34 – continued from previous page

Name	Symbol	Type	Initial Value
PIvq_y	$y_{PIvq}$	Algeb	$I_q0 - K_{Pv}V_q$
PIId_y	$y_{PIId}$	Algeb	$K_{Pi}(-I_d + y_{PIvd})$
PIIq_y	$y_{PIIq}$	Algeb	$K_{Pi}(-I_q + y_{PIvq})$
udref	$u_{dref}$	Algeb	$u_{dref0}$
uqref	$u_{qref}$	Algeb	$u_{qref0}$
a	$\theta$	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	$y_{Psen}$	State	$P_e - y_{Psen}$	$T_c$
Qsen_y	$y_{Qsen}$	State	$Q_e - y_{Qsen}$	$T_c$
Psig_y	$y_{Psig}$	State	$P_{aux} + y_{Psen} - y_{Psig}$	$T_{pm}$
Qsig_y	$y_{Qsig}$	State	$Q_{aux} + y_{Qsen} - y_{Qsig}$	$T_{pm}$
PIplim_xi	$xi_{PIplim}$	State	$K_{Iplim}(-y_{Psen} + y_{Psig})$	
PIqlim_xi	$xi_{PIqlim}$	State	$K_{Iqlim}(-y_{Qsen} + y_{Qsig})$	
delta	$\delta$	State	$y_{dw}$	
PIvd_xi	$xi_{PIvd}$	State	$K_{Iv}(-V_d + v_{ref2})$	
PIvq_xi	$xi_{PIvq}$	State	$-K_{Iv}V_q$	
PIId_xi	$xi_{PIId}$	State	$K_{Ii}(-I_d + y_{PIvd})$	
PIIq_xi	$xi_{PIIq}$	State	$K_{Ii}(-I_q + y_{PIvq})$	
udLag_y	$y_{udLag}$	State	$udref - y_{udLag}$	$T_e$
uqLag_y	$y_{uqLag}$	State	$u_{qref} - y_{uqLag}$	$T_e$
ud	$ud$	AliasState	0	
uq	$uq$	AliasState	0	

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
Paux	$P_{aux}$	Algeb	$P_{aux}$
Qaux	$Q_{aux}$	Algeb	$Q_{aux}$
PIplim_y	$y_{PIplim}$	Algeb	$K_{Pplim} (-y_{Psen} + y_{Psig}) + xi_{PIplim} - y_{PIplim}$
PIqlim_y	$y_{PIqlim}$	Algeb	$K_{Pqlim} (-y_{Qsen} + y_{Qsig}) + xi_{PIqlim} - y_{PIqlim}$
vd	$V_d$	Algeb	$V_u \cos(\delta - \theta) - V_d$
vq	$V_q$	Algeb	$-V_u \sin(\delta - \theta) - V_q$
Pe	$P_e$	Algeb	$I_d V_d + I_q V_q - P_e$
Qe	$Q_e$	Algeb	$I_d V_q - I_q V_d - Q_e$
Id	$I_d$	Algeb	$I_d r_a - I_q x_s + V_d - y_{udLag}$
Iq	$I_q$	Algeb	$I_d x_s + I_q r_a + V_q - y_{uqLag}$
dw_x	$x_{dw}$	Algeb	$\omega_{drp} w_0 (y_{PIplim} - y_{Psen}) - x_{dw}$
dw_y	$y_{dw}$	Algeb	$\Delta \omega_{max} z_u^{lim_{dw}} + \Delta \omega_{min} z_l^{lim_{dw}} + x_{dw} z_i^{lim_{dw}} - y_{dw}$
vref2	$v_{ref2}$	Algeb	$Q_{drp} (u y_{PIqlim} - y_{Qsen}) + V_{ref} - v_{ref2}$
PIvd_y	$y_{PIvd}$	Algeb	$K_{Pv} (-V_d + v_{ref2}) + xi_{PIvd} - y_{PIvd}$
PIqv_y	$y_{PIqv}$	Algeb	$-K_{Pv} V_q + xi_{PIqv} - y_{PIqv}$
PIId_y	$y_{PIId}$	Algeb	$K_{Pi} (-I_d + y_{PIvd}) + xi_{PIId} - y_{PIId}$
PIIq_y	$y_{PIIq}$	Algeb	$K_{Pi} (-I_q + y_{PIqv}) + xi_{PIIq} - y_{PIIq}$
udref	$u_{dref}$	Algeb	$I_d r_a - I_q x_s + V_d - u_{dref} + y_{PIId}$
uqref	$u_{qref}$	Algeb	$I_d x_s + I_q r_a + V_q - u_{qref} + y_{PIIq}$
a	$\theta$	ExtAlgeb	$-P_e u$
v	$V$	ExtAlgeb	$-Q_e u$
Idref	$Idref$	AliasAlgeb	0
Iqref	$Iqref$	AliasAlgeb	0

## Services

Name	Symbol	Equation	Type
Pref	$P_{ref}$	$P_{0s} \gamma_P$	ConstService
Qref	$Q_{ref}$	$Q_{0s} \gamma_Q$	ConstService
w0	$w_0$	$2\pi f$	ConstService
ixf	$1/x_f$	$\frac{1}{x_s}$	ConstService
Id0	$I_{d0}$	$\frac{P_{ref} u}{V}$	ConstService
Iq0	$I_{q0}$	$-\frac{Q_{ref} u}{V}$	ConstService
vd0	$v_{d0}$	$V_u$	ConstService
vq0	$v_{q0}$	0	ConstService
udref0	$u_{dref0}$	$I_{d0} r_a - I_{q0} x_s + v_{d0}$	ConstService
uqref0	$u_{qref0}$	$I_{d0} x_s + I_{q0} r_a + v_{q0}$	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	

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)

## 5.24.6 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	$\omega_{drp}$	frequency droop percentage	0.033		
Qdrp	$Q_{drp}$	Voltage droop percentage	0.045		
Tr	$T_c$	transducer time constant	0.005	s	
Te	$T_e$	output 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	$\gamma_P$	P ratio of linked static gen	1		
gammaq	$\gamma_Q$	Q ratio of linked static gen	1		
mf	$M_f$	VSM inertia constant	0.150		
dd	$d_d$	VSM damping factor	0.110		
pll		PLL device idx (optional)			

## Variables

Name	Symbol	Type	Description	Unit	Properties
Psen_y	$y_{Psen}$	State	State in lag transfer function	v_str	
Qsen_y	$y_{Qsen}$	State	State in lag transfer function	v_str	
Psig_y	$y_{Psig}$	State	State in lag TF	v_str	
Qsig_y	$y_{Qsig}$	State	State in lag TF	v_str	
PIplim_xi	$xi_{PIplim}$	State	Integrator output	v_str	
PIqlim_xi	$xi_{PIqlim}$	State	Integrator output	v_str	
delta	$\delta$	State	virtual delta	rad	v_str
INTw_y	$y_{INTw}$	State	Integrator output	v_str	
PIvd_xi	$xi_{PIvd}$	State	Integrator output	v_str	
PIvq_xi	$xi_{PIvq}$	State	Integrator output	v_str	
PIId_xi	$xi_{PIId}$	State	Integrator output	v_str	
PIIq_xi	$xi_{PIIq}$	State	Integrator output	v_str	
udLag_y	$y_{udLag}$	State	State in lag transfer function	v_str	
uqLag_y	$y_{uqLag}$	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	$y_{PIplim}$	Algeb	PI output	v_str	
PIqlim_y	$y_{PIqlim}$	Algeb	PI output	v_str	
vd	$V_d$	Algeb	d-axis voltage	v_str	
vq	$V_q$	Algeb	q-axis voltage	v_str	
Pe	$P_e$	Algeb	active power injection from VSC	v_str	
Qe	$Q_e$	Algeb	reactive power injection from VSC	v_str	
Id	$I_d$	Algeb	d-axis current	v_str	
Iq	$I_q$	Algeb	q-axis current	v_str	
wref	$\omega_{ref}$	Algeb	speed ref	pu	v_str
dw_x	$x_{dw}$	Algeb	Value before limiter	v_str	
dw_y	$y_{dw}$	Algeb	Output after limiter and post gain	v_str	
vref2	$v_{ref2}$	Algeb	voltage reference after droop	v_str	
PIvd_y	$y_{PIvd}$	Algeb	PI output	v_str	
PIvq_y	$y_{PIvq}$	Algeb	PI output	v_str	
PIId_y	$y_{PIId}$	Algeb	PI output	v_str	
PIIq_y	$y_{PIIq}$	Algeb	PI output	v_str	
udref	$ud_{ref}$	Algeb	ud reference	v_str	
uqref	$uq_{ref}$	Algeb	uq reference	v_str	
a	$\theta$	ExtAlgeb	Bus voltage angle		
v	$V$	ExtAlgeb	Bus voltage magnitude		
plldw	$\Delta\omega_{PLL}$	ExtAlgeb	PLL measured freq. deviation		
Idref	$Id_{ref}$	AliasAlgeb	Alias of PIvd_y		
Iqref	$Iq_{ref}$	AliasAlgeb	Alias of PIvq_y		

## Initialization Equations

Name	Symbol	Type	Initial Value
Psen_y	$y_{Psen}$	State	$P_e$
Qsen_y	$y_{Qsen}$	State	$Q_e$
Psig_y	$y_{Psig}$	State	$P_{aux} + y_{Psen}$
Qsig_y	$y_{Qsig}$	State	$Q_{aux} + y_{Qsen}$
PIplim_xi	$xi_{PIplim}$	State	$y_{Psen}$
PIqlim_xi	$xi_{PIqlim}$	State	$y_{Qsen}$
delta	$\delta$	State	$\theta$
INTw_y	$y_{INTw}$	State	$\omega_{ref}$
PIvd_xi	$xi_{PIvd}$	State	$I_{d0}$
PIvq_xi	$xi_{PIvq}$	State	$I_{q0}$
PIId_xi	$xi_{PIId}$	State	0.0
PIIq_xi	$xi_{PIIq}$	State	0.0
udLag_y	$y_{udLag}$	State	$u_{dref}$
uqLag_y	$y_{uqLag}$	State	$u_{qref}$
ud	$ud$	AliasState	
uq	$uq$	AliasState	
Paux	$P_{aux}$	Algeb	0
Qaux	$Q_{aux}$	Algeb	0
PIplim_y	$y_{PIplim}$	Algeb	$K_{Pplim}(-y_{Psen} + y_{Psig}) + y_{Psen}$
PIqlim_y	$y_{PIqlim}$	Algeb	$K_{Pqlim}(-y_{Qsen} + y_{Qsig}) + y_{Qsen}$
vd	$V_d$	Algeb	$v_{d0}$
vq	$V_q$	Algeb	$v_{q0}$
Pe	$P_e$	Algeb	$P_{ref}$
Qe	$Q_e$	Algeb	$Q_{ref}$
Id	$I_d$	Algeb	$I_{d0}$
Iq	$I_q$	Algeb	$I_{q0}$
wref	$\omega_{ref}$	Algeb	1
dw_x	$x_{dw}$	Algeb	$w_0(-\omega_{ref} + y_{INTw})$
dw_y	$y_{dw}$	Algeb	$\Delta\omega_{max}z_u^{lim_{dw}} + \Delta\omega_{min}z_l^{lim_{dw}} + x_{dw}z_i^{lim_{dw}}$
vref2	$v_{ref2}$	Algeb	$V_{ref}u$
PIvd_y	$y_{PIvd}$	Algeb	$I_{d0} + K_{Pv}(-V_d + v_{ref2})$
PIvq_y	$y_{PIvq}$	Algeb	$I_{q0} - K_{Pv}V_q$
PIId_y	$y_{PIId}$	Algeb	$K_{Pi}(-I_d + y_{PIvd})$
PIIq_y	$y_{PIIq}$	Algeb	$K_{Pi}(-I_q + y_{PIvq})$
udref	$u_{dref}$	Algeb	$u_{dref0}$
uqref	$u_{qref}$	Algeb	$u_{qref0}$
a	$\theta$	ExtAlgeb	
v	$V$	ExtAlgeb	
plldw	$\Delta\omega_{PLL}$	ExtAlgeb	
Idref	$Idref$	AliasAlgeb	
Iqref	$Iqref$	AliasAlgeb	

## Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
Psen_y	$y_{Psen}$	State	$P_e - y_{Psen}$	$T_c$
Qsen_y	$y_{Qsen}$	State	$Q_e - y_{Qsen}$	$T_c$
Psig_y	$y_{Psig}$	State	$P_{aux} + y_{Psen} - y_{Psig}$	$T_{pm}$
Qsig_y	$y_{Qsig}$	State	$Q_{aux} + y_{Qsen} - y_{Qsig}$	$T_{pm}$
PI-plim_xi	$xi_{PIplim}$	State	$K_{Iplim} (-y_{Psen} + y_{Psig})$	
PIqlim_xi	$xi_{PIqlim}$	State	$K_{Iqlim} (-y_{Qsen} + y_{Qsig})$	
delta	$\delta$	State	$y_{dw}$	
INTw_y	$y_{INTw}$	State	$\Delta\omega_{PLL}\omega_{drp}dd + \omega_{drp} (y_{PIplim} - y_{Psen}) + \omega_{ref} - y_{INTw}$	$T_{int}$
PIvd_xi	$xi_{PIvd}$	State	$K_{Iv} (-V_d + v_{ref2})$	
PIvq_xi	$xi_{PIvq}$	State	$-K_{Iv} V_q$	
PIId_xi	$xi_{PIId}$	State	$K_{Ii} (-I_d + y_{PIvd})$	
PIIq_xi	$xi_{PIIq}$	State	$K_{Ii} (-I_q + y_{PIvq})$	
udLag_y	$y_{udLag}$	State	$u_{dref} - y_{udLag}$	$T_e$
uqLag_y	$y_{uqLag}$	State	$u_{qref} - y_{uqLag}$	$T_e$
ud	$ud$	AliasState	0	
uq	$uq$	AliasState	0	

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
Paux	$Paux$	Algeb	$Paux$
Qaux	$Qaux$	Algeb	$Qaux$
PIplim_y	$yPIplim$	Algeb	$K_{Pplim} (-y_{Psen} + y_{Psig}) + xi_{PIplim} - y_{PIplim}$
PIqlim_y	$yPIqlim$	Algeb	$K_{Pqlim} (-y_{Qsen} + y_{Qsig}) + xi_{PIqlim} - y_{PIqlim}$
vd	$V_d$	Algeb	$V_u \cos(\delta - \theta) - V_d$
vq	$V_q$	Algeb	$-V_u \sin(\delta - \theta) - V_q$
Pe	$P_e$	Algeb	$I_d V_d + I_q V_q - P_e$
Qe	$Q_e$	Algeb	$I_d V_q - I_q V_d - Q_e$
Id	$I_d$	Algeb	$I_d r_a - I_q x_s + V_d - y_{udLag}$
Iq	$I_q$	Algeb	$I_d x_s + I_q r_a + V_q - y_{uqLag}$
wref	$\omega_{ref}$	Algeb	$1 - \omega_{ref}$
dw_x	$x_{dw}$	Algeb	$w_0 (-\omega_{ref} + y_{INTw}) - x_{dw}$
dw_y	$y_{dw}$	Algeb	$\Delta \omega_{max} z_u^{lim_{dw}} + \Delta \omega_{min} z_l^{lim_{dw}} + x_{dw} z_i^{lim_{dw}} - y_{dw}$
vref2	$v_{ref2}$	Algeb	$Q_{drp} (u y_{PIqlim} - y_{Qsen}) + V_{ref} - v_{ref2}$
PIvd_y	$yPIvd$	Algeb	$K_{Pv} (-V_d + v_{ref2}) + xi_{PIvd} - y_{PIvd}$
PIvq_y	$yPIvq$	Algeb	$-K_{Pv} V_q + xi_{PIvq} - y_{PIvq}$
PIId_y	$yPIId$	Algeb	$K_{Pi} (-I_d + y_{PIvd}) + xi_{PIId} - y_{PIId}$
PIIq_y	$yPIIq$	Algeb	$K_{Pi} (-I_q + y_{PIvq}) + xi_{PIIq} - y_{PIIq}$
udref	$u_{dref}$	Algeb	$I_d r_a - I_q x_s + V_d - u_{dref} + y_{PIId}$
uqref	$u_{qref}$	Algeb	$I_d x_s + I_q r_a + V_q - u_{qref} + y_{PIIq}$
a	$\theta$	ExtAlgeb	$-P_e u$
v	$V$	ExtAlgeb	$-Q_e u$
plldw	$\Delta \omega_{PLL}$	ExtAlgeb	0
Idref	$Idref$	AliasAlgeb	0
Iqref	$Iqref$	AliasAlgeb	0

## Services

Name	Symbol	Equation	Type
Pref	$P_{ref}$	$P_{0s} \gamma_P$	ConstService
Qref	$Q_{ref}$	$Q_{0s} \gamma_Q$	ConstService
w0	$w_0$	$2\pi f$	ConstService
ixf	$1/x_f$	$\frac{1}{x_s}$	ConstService
Id0	$I_{d0}$	$\frac{P_{ref} u}{V}$	ConstService
Iq0	$I_{q0}$	$-\frac{Q_{ref} u}{V}$	ConstService
vd0	$v_{d0}$	$V_u$	ConstService
vq0	$v_{q0}$	0	ConstService
Tint	$T_{int}$	$M_f \omega_{drp}$	ConstService
udref0	$u_{dref0}$	$I_{d0} r_a - I_{q0} x_s + v_{d0}$	ConstService
uqref0	$u_{qref0}$	$I_{d0} x_s + I_{q0} r_a + v_{q0}$	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	

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)

## 5.24.7 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	$\omega_{drp}$	frequency droop percentage	0.033		
Qdrp	$Q_{drp}$	Voltage droop percentage	0.045		
Tr	$T_c$	transducer time constant	0.005	s	
Te	$T_e$	output 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	$\gamma_P$	P ratio of linked static gen	1		
gammaq	$\gamma_Q$	Q ratio of linked static gen	1		

## Variables

Name	Symbol	Type	Description	Unit	Properties
Psen_y	$y_{Psen}$	State	State in lag transfer function	v_str	
Qsen_y	$y_{Qsen}$	State	State in lag transfer function	v_str	
Psig_y	$y_{Psig}$	State	State in lag TF	v_str	
Qsig_y	$y_{Qsig}$	State	State in lag TF	v_str	
PIplim_xi	$x_{iPIplim}$	State	Integrator output	v_str	
PIqlim_xi	$x_{iPIqlim}$	State	Integrator output	v_str	
delta	$\delta$	State	virtual delta	rad	v_str
vref2	$vref_2$	State		v_str	
PIvd_xi	$x_{iPIvd}$	State	Integrator output	v_str	
PIvq_xi	$x_{iPIvq}$	State	Integrator output	v_str	
PIId_xi	$x_{iPIId}$	State	Integrator output	v_str	
PIIq_xi	$x_{iPIIq}$	State	Integrator output	v_str	
udLag_y	$y_{udLag}$	State	State in lag transfer function	v_str	
uqLag_y	$y_{uqLag}$	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	$y_{PIplim}$	Algeb	PI output	v_str	
PIqlim_y	$y_{PIqlim}$	Algeb	PI output	v_str	
vd	$V_d$	Algeb	d-axis voltage	v_str	
vq	$V_q$	Algeb	q-axis voltage	v_str	
Pe	$P_e$	Algeb	active power injection from VSC	v_str	
Qe	$Q_e$	Algeb	reactive power injection from VSC	v_str	
Id	$I_d$	Algeb	d-axis current	v_str	
Iq	$I_q$	Algeb	q-axis current	v_str	
wref	$\omega_{ref}$	Algeb	speed ref	pu	v_str
Vref	$Vref$	Algeb		v_str	
dw_x	$x_{dw}$	Algeb	Value before limiter	v_str	
dw_y	$y_{dw}$	Algeb	Output after limiter and post gain	v_str	
derv	$dV$	Algeb	input to voltage integrator	v_str	
PIvd_y	$y_{PIvd}$	Algeb	PI output	v_str	
PIvq_y	$y_{PIvq}$	Algeb	PI output	v_str	
PIId_y	$y_{PIId}$	Algeb	PI output	v_str	
PIIq_y	$y_{PIIq}$	Algeb	PI output	v_str	
udref	$u_{dref}$	Algeb	ud reference	v_str	
uqref	$u_{qref}$	Algeb	uq reference	v_str	
a	$\theta$	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	$y_{Psen}$	State	$P_e$
Qsen_y	$y_{Qsen}$	State	$Q_e$
Psig_y	$y_{Psig}$	State	$P_{aux} + y_{Psen}$
Qsig_y	$y_{Qsig}$	State	$Q_{aux} + y_{Qsen}$
PIplim_xi	$xi_{PIplim}$	State	$y_{Psen}$
PIqlim_xi	$xi_{PIqlim}$	State	$y_{Qsen}$
delta	$\delta$	State	$\theta$
vref2	$vref_2$	State	$V_d$
PIvd_xi	$xi_{PIvd}$	State	$I_{d0}$
PIvq_xi	$xi_{PIvq}$	State	$I_{q0}$
PIId_xi	$xi_{PIId}$	State	0.0
PIIq_xi	$xi_{PIIq}$	State	0.0
udLag_y	$y_{udLag}$	State	$u_{dref}$
uqLag_y	$y_{uqLag}$	State	$u_{qref}$
ud	$ud$	AliasState	
uq	$uq$	AliasState	
Paux	$P_{aux}$	Algeb	0
Qaux	$Q_{aux}$	Algeb	0
PIplim_y	$y_{PIplim}$	Algeb	$K_{Plim}(-y_{Psen} + y_{Psig}) + y_{Psen}$
PIqlim_y	$y_{PIqlim}$	Algeb	$K_{qlim}(-y_{Qsen} + y_{Qsig}) + y_{Qsen}$
vd	$V_d$	Algeb	$v_{d0}$
vq	$V_q$	Algeb	$v_{q0}$
Pe	$P_e$	Algeb	$P_{ref}$
Qe	$Q_e$	Algeb	$Q_{ref}$
Id	$I_d$	Algeb	$I_{d0}$
Iq	$I_q$	Algeb	$I_{q0}$
wref	$\omega_{ref}$	Algeb	1
Vref	$V_{ref}$	Algeb	$V_{ref}$
dw_x	$x_{dw}$	Algeb	$\frac{\omega_{drp} w_0 (y_{PIplim} - y_{Psen})}{vref_2^2}$
dw_y	$y_{dw}$	Algeb	$\Delta\omega_{max} z_u^{lim_{dw}} + \Delta\omega_{min} z_l^{lim_{dw}} + x_{dw} z_i^{lim_{dw}}$
derv	$dV$	Algeb	0
PIvd_y	$y_{PIvd}$	Algeb	$I_{d0} + K_{Pv}(-V_d + vref_2)$
PIvq_y	$y_{PIvq}$	Algeb	$I_{q0} - K_{Pv}V_q$
PIId_y	$y_{PIId}$	Algeb	$K_{Pi}(-I_d + y_{PIvd})$
PIIq_y	$y_{PIIq}$	Algeb	$K_{Pi}(-I_q + y_{PIvq})$
udref	$u_{dref}$	Algeb	$u_{dref0}$
uqref	$u_{qref}$	Algeb	$u_{qref0}$
a	$\theta$	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	$y_{Psen}$	State	$P_e - y_{Psen}$	$T_c$
Qsen_y	$y_{Qsen}$	State	$Q_e - y_{Qsen}$	$T_c$
Psig_y	$y_{Psig}$	State	$P_{aux} + y_{Psen} - y_{Psig}$	$T_{pm}$
Qsig_y	$y_{Qsig}$	State	$Q_{aux} + y_{Qsen} - y_{Qsig}$	$T_{pm}$
PIplim_xi	$x_{Iplim}$	State	$K_{Iplim} (-y_{Psen} + y_{Psig})$	
PIqlim_xi	$x_{Iqlim}$	State	$K_{Iqlim} (-y_{Qsen} + y_{Qsig})$	
delta	$\delta$	State	$y_{dw}$	
vref2	$vref_2$	State	$dVw_0$	
PIvd_xi	$x_{Ivd}$	State	$K_{Iv} (-V_d + vref_2)$	
PIvq_xi	$x_{Ivq}$	State	$-K_{Iv} V_q$	
PIId_xi	$x_{IIId}$	State	$K_{II} (-I_d + y_{Ivd})$	
PIIq_xi	$x_{IIq}$	State	$K_{II} (-I_q + y_{Ivq})$	
udLag_y	$y_{udLag}$	State	$udref - y_{udLag}$	$T_e$
uqLag_y	$y_{uqLag}$	State	$uqref - y_{uqLag}$	$T_e$
ud	$ud$	AliasState	0	
uq	$uq$	AliasState	0	

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
Paux	$Paux$	Algeb	$Paux$
Qaux	$Qaux$	Algeb	$Qaux$
PI-plim_y	$yPIplim$	Algeb	$K_{Pplim}(-yPsen + yPsig) + xi_{PIplim} - yPIplim$
PIqlim_y	$yPIqlim$	Algeb	$K_{Pqlim}(-yQsen + yQsig) + xi_{PIqlim} - yPIqlim$
vd	$V_d$	Algeb	$V_u \cos(\delta - \theta) - V_d$
vq	$V_q$	Algeb	$-V_u \sin(\delta - \theta) - V_q$
Pe	$P_e$	Algeb	$I_d V_d + I_q V_q - P_e$
Qe	$Q_e$	Algeb	$I_d V_q - I_q V_d - Q_e$
Id	$I_d$	Algeb	$I_d r_a - I_q x_s + V_d - y_{udLag}$
Iq	$I_q$	Algeb	$I_d x_s + I_q r_a + V_q - y_{uqLag}$
wref	$\omega_{ref}$	Algeb	$1 - \omega_{ref}$
Vref	$V_{ref}$	Algeb	$V_{ref} - Vref$
dw_x	$x_{dw}$	Algeb	$\frac{\omega_{drp} w_0 (y_{PIplim} - y_{Psen})}{vref_2^2} - x_{dw}$
dw_y	$y_{dw}$	Algeb	$\Delta \omega_{max} z_u^{lim_{dw}} + \Delta \omega_{min} z_l^{lim_{dw}} + x_{dw} z_i^{lim_{dw}} - y_{dw}$
derv	$dV$	Algeb	$K_{dvocvref2} (V_{ref} - vref_2) (V_{ref} + vref_2) + \frac{\omega_{drp} (y_{PIqlim} - y_{Qsen})}{vref_2} - dV$
PIvd_y	$yPIvd$	Algeb	$K_{Pv} (-V_d + vref_2) + xi_{PIvd} - yPIvd$
PIqv_y	$yPIqv$	Algeb	$-K_{Pv} V_q + xi_{PIqv} - yPIqv$
PIId_y	$yPIId$	Algeb	$K_{Pi} (-I_d + yPIvd) + xi_{PIId} - yPIId$
PIIq_y	$yPIIq$	Algeb	$K_{Pi} (-I_q + yPIqv) + xi_{PIIq} - yPIIq$
udref	$u_{dref}$	Algeb	$I_d r_a - I_q x_s + V_d - u_{dref} + yPIId$
uqref	$u_{qref}$	Algeb	$I_d x_s + I_q r_a + V_q - u_{qref} + yPIIq$
a	$\theta$	ExtAlgeb	$-P_e u$
v	$V$	ExtAlgeb	$-Q_e u$
Idref	$Idref$	AliasAlgeb	0
Iqref	$Iqref$	AliasAlgeb	0

## Services

Name	Symbol	Equation	Type
Pref	$P_{ref}$	$P_{0s}\gamma_P$	ConstService
Qref	$Q_{ref}$	$Q_{0s}\gamma_Q$	ConstService
w0	$w_0$	$2\pi f$	ConstService
ixf	$1/x_f$	$\frac{1}{x_s}$	ConstService
Id0	$I_{d0}$	$\frac{P_{ref}u}{V}$	ConstService
Iq0	$I_{q0}$	$-\frac{Q_{ref}u}{V}$	ConstService
vd0	$v_{d0}$	$Vu$	ConstService
vq0	$v_{q0}$	0	ConstService
Kdvoc	$Kdvoc$	$\frac{400000000\omega_{drp}}{(99990000-2(100-100Q_{drp}))^2}$	ConstService
udref0	$udref_0$	$I_{d0}r_a - I_{q0}x_s + v_{d0}$	ConstService
uqref0	$uqref_0$	$I_{d0}x_s + I_{q0}r_a + v_{q0}$	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	

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)

## 5.25 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](#)

### 5.25.1 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	De-fault	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
Ht-frac	$D_{shaft}$	Turbine inertia fraction (Hturb/H)	0.500		power
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	$\omega_0$	Default speed if not using a torque model	1	p.u.	
reg			0		
Sn	$S_n$		0		

## Variables

Name	Symbol	Type	Description	Unit	Properties
s1_y	$y_{s1}$	State	Integrator output	v_str	
s2_y	$y_{s2}$	State	Integrator output	v_str	
s3_y	$y_{s3}$	State	Integrator output	v_str	
wt	$\omega_t$	AliasState	Alias of s1_y		
wg	$\omega_g$	AliasState	Alias of s2_y		
wr0	$\omega_{r0}$	Algeb	speed set point	p.u.	v_str
Pm	$P_m$	Algeb	Mechanical power		v_str
pd	$P_d$	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	$y_{s1}$	State	$\omega_{r0}$
s2_y	$y_{s2}$	State	$\omega_{r0}$
s3_y	$y_{s3}$	State	$\frac{P_{e0}}{\omega_{r0}}$
wt	$\omega_t$	AliasState	
wg	$\omega_g$	AliasState	
wr0	$\omega_{r0}$	Algeb	$\omega_0$
Pm	$P_m$	Algeb	$P_{e0}$
pd	$P_d$	Algeb	0
wge	$wge$	ExtAlgeb	
Pe	$P_e$	ExtAlgeb	

## Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
s1_y	$y_{s1}$	State	$-1.0P_d + \frac{1.0P_m}{y_{s1}} - 1.0y_{s3}$	$2H_t$
s2_y	$y_{s2}$	State	$-1.0Damp(-\omega_0 + y_{s2}) + 1.0P_d - \frac{1.0P_e}{y_{s2}} + 1.0y_{s3}$	$2H_g$
s3_y	$y_{s3}$	State	$K_{shaft}(y_{s1} - y_{s2})$	1.0
wt	$\omega_t$	AliasState	0	
wg	$\omega_g$	AliasState	0	

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
wr0	$\omega_{r0}$	Algeb	$\omega_0 - \omega_{r0}$
Pm	$P_m$	Algeb	$-P_m + P_{e0}$
pd	$P_d$	Algeb	$D_{shaft}(y_{s1} - y_{s2}) - P_d$
wge	$wge$	ExtAlgeb	$y_{s2} - 1.0$
Pe	$P_e$	ExtAlgeb	0

## Services

Name	Symbol	Equation	Type
Ht2	$2H_t$	$2D_{shaft}H_t$	ConstService
Hg2	$2H_g$	$2H_t(1 - D_{shaft})$	ConstService
Kshaft	$K_{shaft}$	$\frac{0.5 \cdot 2H_g 2H_t F_{red1}^2}{H_t}$	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)

## 5.25.2 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	$\omega_0$	Default speed if not using a torque model	1	p.u.	
reg			0		
Sn	$S_n$		0		

## Variables

Name	Symbol	Type	Description	Unit	Properties
s1_y	$y_{s1}$	State	Integrator output		v_str
s3_y	$y_{s3}$	State	Unused state variable		
wt	$\omega_t$	AliasState	Alias of s1_y		
wg	$\omega_g$	AliasState	Alias of s1_y		
Pm	$P_m$	Algeb	Mechanical power		v_str
wr0	$\omega_{r0}$	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	$y_{s1}$	State	$\omega_{r0}$
s3_y	$y_{s3}$	State	
wt	$\omega_t$	AliasState	
wg	$\omega_g$	AliasState	
Pm	$P_m$	Algeb	$P_{e0}$
wr0	$\omega_{r0}$	Algeb	$\omega_0$
wge	$wge$	ExtAlgeb	
Pe	$Pe$	ExtAlgeb	

## Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
s1_y	$y_{s1}$	State	$-1.0D_{shaft}(-\omega_{r0} + y_{s1}) + \frac{1.0(P_m - P_e)}{wge}$	$2H$
s3_y	$y_{s3}$	State	0	
wt	$\omega_t$	AliasState	0	
wg	$\omega_g$	AliasState	0	

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
Pm	$P_m$	Algeb	$-P_m + P_{e0}$
wr0	$\omega_{r0}$	Algeb	$\omega_0 - \omega_{r0}$
wge	$wge$	ExtAlgeb	$y_{s1} - 1.0$
Pe	$P_e$	ExtAlgeb	0

## Services

Name	Symbol	Equation	Type
H2	$2H$	$2H_t$	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)

## 5.26 RenPitch

Renewable generator pitch controller group.

Common Parameters: u, name, rea

Available models: [WTPTA1](#)

### 5.26.1 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	$T_\theta$	Blade response time const.	0.300	s	
thmax	$\theta_{max}$	Max. pitch angle	30	deg.	
thmin	$\theta_{min}$	Min. pitch angle	0	deg.	
dthmax	$\dot{\theta}_{max}$	Max. pitch angle rate	5	deg.	
dthmin	$\dot{\theta}_{min}$	Min. pitch angle rate	-5	deg.	
rego			0		
ree			0		

## Variables

Name	Symbol	Type	Description	Unit	Properties
PIc_xi	$xi_{PI_c}$	State	Integrator output	v_str	
PIw_xi	$xi_{PI_w}$	State	Integrator output	v_str	
LG_y	$y_{LG}$	State	State in lag TF	v_str	
wt	$wt$	ExtState			
Pord	$Pord$	ExtState			
PIc_yul	$y_{PI_c}^{ul}$	Algeb		v_str	
PIc_y	$y_{PI_c}$	Algeb	PI output	v_str	
wref	$\omega_{ref}$	Algeb	optional speed reference	v_str	
PIw_yul	$y_{PI_w}^{ul}$	Algeb		v_str	
PIw_y	$y_{PI_w}$	Algeb	PI output	v_str	
theta	$\theta$	ExtAlgeb			
Pref	$Pref$	ExtAlgeb			

## Initialization Equations

Name	Symbol	Type	Initial Value
PIc_xi	$xi_{PI_c}$	State	0.0
PIw_xi	$xi_{PI_w}$	State	0.0
LG_y	$y_{LG}$	State	$1.0y_{PI_c} + 1.0y_{PI_w}$
wt	$wt$	ExtState	
Pord	$Pord$	ExtState	
PIc_yul	$y_{PI_c}^{ul}$	Algeb	$K_{pc}(Pord - Pref)$
PIc_y	$y_{PI_c}$	Algeb	$\theta_{max}z_u^{hl_{PI_c}} + \theta_{min}z_l^{hl_{PI_c}} + y_{PI_c}^{ul}z_i^{hl_{PI_c}}$
wref	$\omega_{ref}$	Algeb	$wt$
PIw_yul	$y_{PI_w}^{ul}$	Algeb	$K_{pw}(K_{cc}(Pord - Pref) - \omega_{ref} + wt)$
PIw_y	$y_{PI_w}$	Algeb	$\theta_{max}z_u^{hl_{PI_w}} + \theta_{min}z_l^{hl_{PI_w}} + y_{PI_w}^{ul}z_i^{hl_{PI_w}}$
theta	$\theta$	ExtAlgeb	
Pref	$Pref$	ExtAlgeb	

## Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
PIc_xi	$xi_{PI_c}$	State	$K_{ic}(Pord - Pref)$	
PIw_xi	$xi_{PI_w}$	State	$K_{iw}(K_{cc}(Pord - Pref) - \omega_{ref} + wt)$	
LG_y	$y_{LG}$	State	$-y_{LG} + 1.0y_{PI_c} + 1.0y_{PI_w}$	$T_\theta$
wt	$wt$	ExtState	0	
Pord	$Pord$	ExtState	0	

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
PIc_yul	$y_{PI_c}^{ul}$	Algeb	$K_{pc}(P_{ord} - P_{ref}) + xi_{PI_c} - y_{PI_c}^{ul}$
PIc_y	$y_{PI_c}$	Algeb	$\theta_{max}z_u^{hl_{PI_c}} + \theta_{min}z_l^{hl_{PI_c}} + y_{PI_c}^{ul}z_i^{hl_{PI_c}} - y_{PI_c}$
wref	$\omega_{ref}$	Algeb	$-\omega_{ref} + wref_0$
PIw_yul	$y_{PI_w}^{ul}$	Algeb	$K_{pw}(K_{cc}(P_{ord} - P_{ref}) - \omega_{ref} + wt) + xi_{PI_w} - y_{PI_w}^{ul}$
PIw_y	$y_{PI_w}$	Algeb	$\theta_{max}z_u^{hl_{PI_w}} + \theta_{min}z_l^{hl_{PI_w}} + y_{PI_w}^{ul}z_i^{hl_{PI_w}} - y_{PI_w}$
theta	$\theta$	ExtAlgeb	$-\theta_0 + y_{LG}$
Pref	$P_{ref}$	ExtAlgeb	0

## Services

Name	Symbol	Equation	Type
wref0	$wref_0$	$\omega_{ref}$	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)

## 5.27 RenPlant

Renewable plant control group.

Common Parameters: u, name

Available models: [REPCA1](#)

### 5.27.1 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.

Notes for PSS/E DYR parser:

1. If ICONs M+1 and M+2 are set to 0 when using generator power, an error will be thrown by the parser, saying "<REPCA1> cannot retrieve <bus1> from <ACLine> using <line>: KeyError('Group <ACLine> does not contain device with idx=False')". Manual effort is required to run the converted file. In the REPCA1 sheet, provide the idx of a line that connects to the RenGen bus.
2. PSS/E enters ICONs M+3 as a string in single quotes. The pair of single quotes need to be removed, or the conversion will fail.

### 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			mandatory
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

continues on next page

Table 41 – continued from previous page

Name	Symbol	Description	Default	Unit	Properties
PLflag		Pline ctrl. flag; 0-disable, 1-enable	True	<i>bool</i>	
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	$f_{fdbd1}$	Lower threshold for freq. error deadband	-0.000	<i>p.u. (Hz)</i>	
fdbd2	$f_{fdbd2}$	Upper threshold for freq. error deadband	0.000	<i>p.u. (Hz)</i>	
femax	$f_{emax}$	Upper limit for freq. error	0.050		
femin	$f_{emin}$	Lower limit for freq. error	-0.050		
Pmax	$P_{max}$	Upper limit on power error (used by PI ctrl.)	999	<i>p.u. (MW)</i>	power
Pmin	$P_{min}$	Lower limit on power error (used by PI ctrl.)	-999	<i>p.u. (MW)</i>	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		
reg		Retrieved RenGen idx			
bus		Retrieved bus idx			
bus1		Retrieved Line.bus1 idx			
bus2		Retrieved Line.bus2 idx			
r		Retrieved Line.r			
x		Retrieved Line.x			

## Variables

Name	Symbol	Type	Description	Unit	Properties
s0_y	$y_{s_0}$	State	State in lag transfer function		v_str
s1_y	$y_{s_1}$	State	State in lag transfer function		v_str
s2_xi	$xi_{s_2}$	State	Integrator output		v_str

continues on next page

Table 42 – continued from previous page

Name	Symbol	Type	Description	Unit	Properties
s3_x	$x'_{s_3}$	State	State in lead-lag		v_str
s4_y	$y_{s_4}$	State	State in lag transfer function		v_str
s5_xi	$xi_{s_5}$	State	Integrator output		v_str
s6_y	$y_{s_6}$	State	State in lag transfer function		v_str
Vref	$Q_{ref}$	Algeb			v_str
Qlinef	$Q_{linef}$	Algeb			v_str
Refsel	$R_{efsel}$	Algeb			v_str
dbd_y	$y_{dbd}$	Algeb	Deadband type 1 output		v_str
enf	$e_{nf}$	Algeb	e Hardlimit output before freeze		v_str
s2_ys	$y_{ss_2}$	Algeb	PI summation before limit		v_str
s2_y	$y_{s_2}$	Algeb	PI output		v_str
s3_y	$y_{s_3}$	Algeb	Output of lead-lag		v_str
ferr	$f_{err}$	Algeb	Frequency deviation	p.u. (Hz)	v_str
fdbd_y	$y_{fdbd}$	Algeb	Deadband type 1 output		v_str
Plant_pref	$P_{ref}$	Algeb	Plant P ref		v_str
Plerr	$P_{lerr}$	Algeb	Pline error		v_str
Perr	$P_{err}$	Algeb	Power error before fe limits		v_str
s5_ys	$y_{ss_5}$	Algeb	PI summation before limit		v_str
s5_y	$y_{s_5}$	Algeb	PI output		v_str
Pext	$P_{ext}$	ExtAlgeb	Pref from RenExciter renamed as Pext		
Qext	$Q_{ext}$	ExtAlgeb	Qref from RenExciter renamed as Qext		
v	$V$	ExtAlgeb	Bus (or busr, if given) terminal voltage		
a	$\theta$	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	$\theta_1$	ExtAlgeb	Angle at Line.bus1		
a2	$\theta_2$	ExtAlgeb	Angle at Line.bus2		

## Initialization Equations

Name	Symbol	Type	Initial Value
s0_y	$y_{s_0}$	State	$V_{comp}s_1^{SW_{VC}} + s_0^{SW_{VC}} (K_c Q_{line} + V)$
s1_y	$y_{s_1}$	State	$Q_{line}$
s2_xi	$xi_{s_2}$	State	0.0
s3_x	$x'_{s_3}$	State	$y_{s_2}$
s4_y	$y_{s_4}$	State	$P_{line}$
s5_xi	$xi_{s_5}$	State	0.0
s6_y	$y_{s_6}$	State	$y_{s_5}$
Vref	$Q_{ref}$	Algeb	$V_{ref0}$
Qlinef	$Q_{linef}$	Algeb	$Q_{line0}$

continues on next page

Table 43 – continued from previous page

Name	Symbol	Type	Initial Value
Refsel	$R_{refsel}$	Algeb	$s_0^{SW_{Ref}} (Q_{linef} - y_{s_1}) + s_1^{SW_{Ref}} (Q_{ref} - y_{s_0})$
dbd_y	$y_{dbd}$	Algeb	$1.0z_l^{db_{dbd}} (R_{refsel} - d_{bd1}) + 1.0z_u^{db_{dbd}} (R_{refsel} - d_{bd2})$
enf	$e_{nf}$	Algeb	$e_{max}z_u^{e_{HL}} + e_{min}z_l^{e_{HL}} + y_{dbd}z_i^{e_{HL}}$
s2_ys	$y_{s_2}$	Algeb	$K_p e_{hld}$
s2_y	$y_{s_2}$	Algeb	$Q_{max}z_u^{lim_{s_2}} + Q_{min}z_l^{lim_{s_2}} + y_{s_2}z_i^{lim_{s_2}}$
s3_y	$y_{s_3}$	Algeb	$y_{s_2}$
ferr	$f_{err}$	Algeb	$-f + f_{ref}$
fdbd_y	$y_{fdbd}$	Algeb	$1.0z_l^{db_{fdbd}} (-f_{dbd1} + f_{err}) + 1.0z_u^{db_{fdbd}} (-f_{dbd2} + f_{err})$
Plant_pref	$P_{ref}$	Algeb	$P_{line0}$
Plerr	$P_{lerr}$	Algeb	$P_{ref} - y_{s_4}$
Perr	$P_{err}$	Algeb	$D_{dn}y_{fdbd}z_1^{f_{dlt0}} + D_{up}y_{fdbd}z_0^{f_{dlt0}} + P_{lerr}s_1^{SW_{PL}}$
s5_ys	$y_{s_5}$	Algeb	$K_{pg} \left( P_{err}z_i^{f_{eHL}} + f_{emax}z_u^{f_{eHL}} + f_{emin}z_l^{f_{eHL}} \right)$
s5_y	$y_{s_5}$	Algeb	$P_{max}z_u^{lim_{s_5}} + P_{min}z_l^{lim_{s_5}} + y_{s_5}z_i^{lim_{s_5}}$
Pext	$P_{ext}$	ExtAlgeb	
Qext	$Q_{ext}$	ExtAlgeb	
v	$V$	ExtAlgeb	
a	$\theta$	ExtAlgeb	
f	$f$	ExtAlgeb	
v1	$V_1$	ExtAlgeb	
v2	$V_2$	ExtAlgeb	
a1	$\theta_1$	ExtAlgeb	
a2	$\theta_2$	ExtAlgeb	

## Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
s0_y	$y_{s_0}$	State	$V_{comp}s_1^{SW_{VC}} + s_0^{SW_{VC}} (K_c Q_{line} + V) - y_{s_0}$	$T_{fltr}$
s1_y	$y_{s_1}$	State	$Q_{line} - y_{s_1}$	$T_{fltr}$
s2_xi	$xi_{s_2}$	State	$K_i (e_{hld} + 2y_{s_2} - 2y_{s_2})$	
s3_x	$x'_{s_3}$	State	$-x'_{s_3} + y_{s_2}$	$T_{fv}$
s4_y	$y_{s_4}$	State	$P_{line} - y_{s_4}$	$T_p$
s5_xi	$xi_{s_5}$	State	$K_{ig} \left( P_{err}z_i^{f_{eHL}} + f_{emax}z_u^{f_{eHL}} + f_{emin}z_l^{f_{eHL}} + 2y_{s_5} - 2y_{s_5} \right)$	
s6_y	$y_{s_6}$	State	$y_{s_5} - y_{s_6}$	$T_g$

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
Vref	$Q_{ref}$	Algeb	$-Q_{ref} + V_{ref0}$
Qlinef	$Q_{linef}$	Algeb	$Q_{line0} - Q_{linef}$
Refsel	$R_{efsel}$	Algeb	$-R_{efsel} + s_0^{SW_{Ref}} (Q_{linef} - y_{s1}) + s_1^{SW_{Ref}} (Q_{ref} - y_{s0})$
dbd_y	$y_{dbd}$	Algeb	$-y_{dbd} + 1.0z_l^{db_{dbd}} (R_{efsel} - d_{bd1}) + 1.0z_u^{db_{dbd}} (R_{efsel} - d_{bd2})$
enf	$e_{nf}$	Algeb	$e_{max}z_u^{e_{HL}} + e_{min}z_l^{e_{HL}} - e_{nf} + y_{dbd}z_i^{e_{HL}}$
s2_ys	$y_{s2}$	Algeb	$K_{pehld} + xi_{s2} - y_{s2}$
s2_y	$y_{s2}$	Algeb	$Q_{max}z_u^{lim_{s2}} + Q_{min}z_l^{lim_{s2}} - y_{s2} + y_{s2}z_i^{lim_{s2}}$
s3_y	$y_{s3}$	Algeb	$T_{ft}(-x'_{s3} + y_{s2}) + T_{fv}x'_{s3} - T_{fv}y_{s3} + (z_1^{LT_{s3}})^2 (-x'_{s3} + y_{s3})$
ferr	$f_{err}$	Algeb	$-f - f_{err} + f_{ref}$
fdbd_y	$y_{fdbd}$	Algeb	$-y_{fdbd} + 1.0z_l^{db_{fdbd}} (-f_{bd1} + f_{err}) + 1.0z_u^{db_{fdbd}} (-f_{bd2} + f_{err})$
Plant_pref	$P_{ref}$	Algeb	$P_{line0} - P_{ref}$
Plerr	$P_{lerr}$	Algeb	$-P_{lerr} + P_{ref} - y_{s4}$
Perr	$P_{err}$	Algeb	$D_{dn}y_{fdbd}z_1^{f_{dt0}} + D_{up}y_{fdbd}z_0^{f_{dt0}} - P_{err} + P_{lerr}s_1^{SW_{PL}}$
s5_ys	$y_{s5}$	Algeb	$K_{pg} (P_{err}z_i^{f_{eHL}} + f_{emax}z_u^{f_{eHL}} + f_{emin}z_l^{f_{eHL}}) + xi_{s5} - y_{s5}$
s5_y	$y_{s5}$	Algeb	$P_{max}z_u^{lim_{s5}} + P_{min}z_l^{lim_{s5}} - y_{s5} + y_{s5}z_i^{lim_{s5}}$
Pext	$P_{ext}$	ExtAlgeb	$s_1^{SWF} y_{s6}$
Qext	$Q_{ext}$	ExtAlgeb	$y_{s3}$
v	$V$	ExtAlgeb	0
a	$\theta$	ExtAlgeb	0
f	$f$	ExtAlgeb	0
v1	$V_1$	ExtAlgeb	0
v2	$V_2$	ExtAlgeb	0
a1	$\theta_1$	ExtAlgeb	0
a2	$\theta_2$	ExtAlgeb	0

## Services

Name	Symbol	Equation	Type
Isign	$I_{sign}$	0	CurrentSign
Iline	$I_{line}$	$\frac{I_{sign}(V_1 e^{i\theta_1} - V_2 e^{i\theta_2})}{r+ix}$	VarService
Iline0	$I_{line0}$	$I_{line}$	ConstService
Pline	$P_{line}$	$\text{re}\left(I_{sign} V_1 \text{conj}\left(\frac{V_1 e^{i\theta_1} - V_2 e^{i\theta_2}}{r+ix}\right) e^{i\theta_1}\right)$	VarService
Pline0	$P_{line0}$	$P_{line}$	ConstService
Qline	$Q_{line}$	$\text{im}\left(I_{sign} V_1 \text{conj}\left(\frac{V_1 e^{i\theta_1} - V_2 e^{i\theta_2}}{r+ix}\right) e^{i\theta_1}\right)$	VarService
Qline0	$Q_{line0}$	$Q_{line}$	ConstService
Vcomp	$V_{comp}$	$ I_{line}(R_{cs} + iX_{cs}) - V e^{i\theta} $	VarService
Vref0	$V_{ref0}$	$V_{comp} s_1^{SWVC} + s_0^{SWVC} (K_c Q_{line0} + V)$	ConstService
zf	$z_f$	$f_{rz} \text{ Indicator}(V < V_{frz})$	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	$db_{dbd}$	DeadBand	
eHL	$e_{HL}$	Limiter	Hardlimit on deadband output
s2_lim	$lim_{s_2}$	HardLimiter	
s3_LT1	$LT_{s_3}$	LessThan	
s3_LT2	$LT_{s_3}$	LessThan	
fdbd_db	$db_{fdbd}$	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	

## 5.28 RenTorque

Renewable torque (Pref) controller.

Common Parameters: u, name

Available models: [WTTQA1](#)

### 5.28.1 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	<i>bool</i>	
name		device name			
rep		RenPitch controller idx			mandatory
Kip	$K_{ip}$	Pref-control integral gain	0.100	<i>p.u.</i>	
Kpp	$K_{pp}$	Pref-control proportional gain	0	<i>p.u.</i>	
Tp	$T_p$	Pe sensing time const.	0.050	<i>s</i>	
Twref	$T_{wref}$	Speed reference time const.	30	<i>s</i>	
Temax	$T_{emax}$	Max. electric torque	1.200	<i>p.u.</i>	power
Temin	$T_{emin}$	Min. electric torque	0	<i>p.u.</i>	power
Tflag		Tflag; 1-power error, 0-speed error		<i>bool</i>	mandatory
p1	$p_1$	Active power point 1	0.200	<i>p.u.</i>	power
sp1	$s_{p1}$	Speed power point 1	0.580	<i>p.u.</i>	
p2	$p_2$	Active power point 2	0.400	<i>p.u.</i>	power
sp2	$s_{p2}$	Speed power point 2	0.720	<i>p.u.</i>	
p3	$p_3$	Active power point 3	0.600	<i>p.u.</i>	power
sp3	$s_{p3}$	Speed power point 3	0.860	<i>p.u.</i>	
p4	$p_4$	Active power point 4	0.800	<i>p.u.</i>	power
sp4	$s_{p4}$	Speed power point 4	1	<i>p.u.</i>	
Tn	$T_n$	Turbine rating. Use Sn from gov if none.	nan	<i>MVA</i>	
rea			0		
rego			0		
ree			0		
reg			0		
Sngo	$S_{n,go}$		0		
w0	$\omega_0$		0		

## Variables

Name	Symbol	Type	Description	Unit	Properties
s1_y	$y_{s_1}$	State	State in lag transfer function		v_str
s2_y	$y_{s_2}$	State	State in lag transfer function		v_str
PI_xi	$x_{PI}$	State	Integrator output		v_str
wg	$\omega_g$	ExtState			v_str,v.setter
wt	$\omega_t$	ExtState			v_str,v.setter
s3_y	$y_{s3}$	ExtState			v_str,v.setter
fPe_y	$y_{fPe}$	Algeb	Output of piecewise		v_str
Tsel	$T_{sel}$	Algeb	Output after Tflag selector		v_str
PI_yul	$y_{PI}^{ul}$	Algeb			v_str
PI_y	$y_{PI}$	Algeb	PI output		v_str
Pe	$P_e$	ExtAlgeb			
wr0	$\omega_{r0}$	ExtAlgeb	Retrieved initial w0 from RenGovernor		v_str,v.setter
wge	$\omega_{ge}$	ExtAlgeb			v_str,v.setter
Pref	$P_{ref}$	ExtAlgeb			v_str,v.setter

## Initialization Equations

Nam	Sym-	Type	Initial Value	
	bol			
s1_y	$y_{s1}$	State	$1.0P_e$	
s2_y	$y_{s2}$	State	$1.0y_{fPe}$	
PI_xi	$xi_{PI}$	State	$\frac{P_{ref0}}{y_{fPe}}$	
wg	$\omega_g$	ExtSta	$y_{fPe}$	
wt	$\omega_t$	ExtSta	$y_{fPe}$	
s3_y	$y_{s3}$	ExtSta	$\frac{P_{ref0}}{\omega_q}$	
fPe_	$y_{fPe}$	Al- geb	FixPiecewise $((s_{p1}, p_1 \geq y_{s1}), (k_{p1}(-p_1 + y_{s1}) + s_{p1}, p_2 \geq y_{s1}), (k_{p2}(-p_2 + y_{s1}) + s_{p2}, p_3 \geq y_{s1}))$	
Tsel	$T_{sel}$	Al- geb	$s_0^{SW_T} (-\omega_g + y_{s2}) + \frac{s_1^{SW_T} (P_e - P_{ref0})}{\omega_g}$	
PI_yl	$y_{PI}^{ul}$	Al- geb	$K_{pp}T_{sel} + \frac{P_{ref0}}{y_{fPe}}$	
PI_y	$y_{PI}$	Al- geb	$T_{emax}z_u^{hlPI} + T_{emin}z_l^{hlPI} + y_{PI}^{ul}z_i^{hlPI}$	
Pe	$P_e$	Ex- tAl- geb		
wr0	$\omega_{r0}$	Ex- tAl- geb	$y_{fPe}$	
wge	$\omega_{ge}$	Ex- tAl- geb	1.0	
Pref	$P_{ref}$	Ex- tAl- geb	$\omega_g y_{PI}$	

## Differential Equations

Name	Symbol	Type	RHS of Equation "T' x' = f(x, y)"	T (LHS)
s1_y	$y_{s1}$	State	$1.0P_e - y_{s1}$	$T_p$
s2_y	$y_{s2}$	State	$1.0y_{fPe} - y_{s2}$	$T_{wref}$
PI_xi	$xi_{PI}$	State	$K_{ip}T_{sel}$	
wg	$\omega_g$	ExtState	0	
wt	$\omega_t$	ExtState	0	
s3_y	$y_{s3}$	ExtState	0	

## Algebraic Equations

Nam	Sym-	Type	RHS of Equation "0 = g(x, y)"
	bol		
fPe_1	$y_{fPe}$	Al-	$-y_{fPe} + \text{FixPiecewise}((s_{p1}, p_1 \geq y_{s1}), (k_{p1}(-p_1 + y_{s1}) + s_{p1}, p_2 \geq y_{s1}), (k_{p2}(-p_2 + y_{s1}) + s_{p2}, p_3 \geq y_{s1}))$
Tsel	$T_{sel}$	Al-	$-T_{sel} + s_0^{SW_T}(-\omega_g + y_{s2}) + \frac{s_1^{SW_T}(P_e - P_{ref0})}{\omega_g}$
PI_y1	$y_{PI}^{ul}$	Al-	$K_{pp}T_{sel} + xi_{PI} - y_{PI}^{ul}$
PI_y	$y_{PI}$	Al-	$T_{emax}z_u^{hl_{PI}} + T_{emin}z_l^{hl_{PI}} + y_{PI}^{ul}z_i^{hl_{PI}} - y_{PI}$
Pe	$P_e$	Ex-	0
wr0	$\omega_{r0}$	Ex-	$-\omega_0 + y_{fPe}$
wge	$\omega_{ge}$	Ex-	$1 - y_{fPe}$
Pref	$P_{ref}$	Ex-	$-\frac{P_{ref0}}{\omega_{ge}} + \omega_g y_{PI}$

## Services

Name	Symbol	Equation	Type
kp1	$k_{p1}$	$\frac{-s_{p1}+s_{p2}}{-p_1+p_2}$	ConstService
kp2	$k_{p2}$	$\frac{-s_{p2}+s_{p3}}{-p_2+p_3}$	ConstService
kp3	$k_{p3}$	$\frac{-s_{p3}+s_{p4}}{-p_3+p_4}$	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)

## 5.29 StaticACDC

AC DC device for power flow

Common Parameters: u, name

Available models: [VSCShunt](#)

### 5.29.1 VSCShunt

Data for VSC Shunt in power flow

## Parameters

Name	Symbol	Description	De-fault	Unit	Properties
idx		unique device idx			
u	$u$	connection status	1	<i>bool</i>	
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	<i>kV</i>	non_zero
Vdcn2	$V_{dcn2}$	DC voltage rating on node 2	100	<i>kV</i>	non_zero
Idcn	$I_{dcn}$	DC current rating	1	<i>kA</i>	non_zero
rsh	$r_{sh}$	AC interface resistance	0.003	<i>ohm</i>	<i>z</i>
xsh	$x_{sh}$	AC interface reactance	0.060	<i>ohm</i>	<i>z</i>
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	<i>pu</i>	
q0		AC reactive power setting	0	<i>pu</i>	
vdc0	$v_{dc0}$	DC voltage setting	1	<i>pu</i>	
k0		Loss coefficient - constant	0		
k1		Loss coefficient - linear	0		
k2		Loss coefficient - quadratic	0		
droop		Enable dc voltage droop control	0	<i>boolean</i>	
K		Droop coefficient	0		
vhigh		Upper voltage threshold in droop control	9999	<i>pu</i>	
vlow		Lower voltage threshold in droop control	0	<i>pu</i>	
vsh-max		Maximum ac interface voltage	1.100	<i>pu</i>	
vsh-min		Minimum ac interface voltage	0.900	<i>pu</i>	
Ish-max		Maximum ac current	2	<i>pu</i>	

## Variables

Name	Symbol	Type	Description	Unit	Properties
ash	$\theta_{sh}$	Algeb	voltage phase behind the transformer	rad	v_str
vsh	$V_{sh}$	Algeb	voltage magnitude behind transformer	p.u.	v_str
psh	$P_{sh}$	Algeb	active power injection into VSC	p.u.	v_str
qsh	$Q_{sh}$	Algeb	reactive power injection into VSC		v_str
pdc	$P_{dc}$	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	$\theta_{sh}$	Algeb	$a$
vsh	$V_{sh}$	Algeb	$v_0$
psh	$P_{sh}$	Algeb	$p_0 (s_0^{mode} + s_1^{mode})$
qsh	$Q_{sh}$	Algeb	$q_0 (s_0^{mode} + s_2^{mode})$
pdc	$P_{dc}$	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	$\theta_{sh}$	Algeb	$-P_{sh} + u (V_{sh} b_{sh} v \sin(\theta_{sh} - a) - V_{sh} g_{sh} v \cos(\theta_{sh} - a) + g_{sh} v^2)$
vsh	$V_{sh}$	Algeb	$-Q_{sh} + u (V_{sh} b_{sh} v \cos(\theta_{sh} - a) + V_{sh} g_{sh} v \sin(\theta_{sh} - a) - b_{sh} v^2)$
psh	$P_{sh}$	Algeb	$u (-P_{sh} + p_0) (s_0^{mode} + s_1^{mode}) + u (s_2^{mode} + s_3^{mode}) (v_1 - v_2 - v_{dc0})$
qsh	$Q_{sh}$	Algeb	$u (-Q_{sh} + q_0) (s_0^{mode} + s_2^{mode}) + u (s_1^{mode} + s_3^{mode}) (-v + v_0)$
pdc	$P_{dc}$	Algeb	$P_{dc} + u (V_{sh}^2 g_{sh} - V_{sh} b_{sh} v \sin(\theta_{sh} - a) - V_{sh} g_{sh} v \cos(\theta_{sh} - a))$
a	$a$	ExtAlgeb	$-P_{sh}$
v	$v$	ExtAlgeb	$-Q_{sh}$
v1	$v_1$	ExtAlgeb	$-\frac{P_{dc}}{v_1 - v_2}$
v2	$v_2$	ExtAlgeb	$\frac{P_{dc}}{v_1 - v_2}$

## Services

Name	Symbol	Equation	Type
gsh	$g_{sh}$	$\frac{\text{re}(r_{sh}) - \text{im}(x_{sh})}{(\text{re}(r_{sh}) - \text{im}(x_{sh}))^2 + (\text{re}(x_{sh}) + \text{im}(r_{sh}))^2}$	ConstService
bsh	$b_{sh}$	$\frac{-\text{re}(x_{sh}) - \text{im}(r_{sh})}{(\text{re}(r_{sh}) - \text{im}(x_{sh}))^2 + (\text{re}(x_{sh}) + \text{im}(r_{sh}))^2}$	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)

## 5.30 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`, `Sn`, `Vn`, `p0`, `q0`, `ra`, `xs`, `subidx`

Common Variables: `q`, `a`, `v`

Available models: [PV](#), [Slack](#)

### 5.30.1 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.

#### Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	$u$	connection status	1	<i>bool</i>	
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	<i>p.u.</i>	
q0	$q_0$	reactive power set point in system base	0	<i>p.u.</i>	
pmax	$p_{max}$	maximum active power in system base	999	<i>p.u.</i>	
pmin	$p_{min}$	minimum active power in system base	-1	<i>p.u.</i>	
qmax	$q_{max}$	maximum reactive power in system base	999	<i>p.u.</i>	
qmin	$q_{min}$	minimum reactive power in system base	-999	<i>p.u.</i>	
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		
busv0	$V_{0bus}$		0		

## Variables

Name	Symbol	Type	Description	Unit	Properties
q	$q$	Algeb	actual reactive power generation	p.u.	v_str
a	$\theta$	ExtAlgeb			
v	$V$	ExtAlgeb			v_str,v_setter

## Initialization Equations

Name	Symbol	Type	Initial Value
q	$q$	Algeb	$q_0 u$
a	$\theta$	ExtAlgeb	
v	$V$	ExtAlgeb	$V_{0bus} (1 - u) + uv_0$

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
q	$q$	Algeb	$u \left( z_i^{qlim} (-V + v_0) + z_l^{qlim} (-q + q_{min}) + z_u^{qlim} (-q + q_{max}) \right)$
a	$\theta$	ExtAlgeb	$-pu$
v	$V$	ExtAlgeb	$-qu$

## 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
al-low_adjust		1	allow adjusting upper or lower limits	(0, 1)
ad-just_lower		0	adjust lower limit	(0, 1)
ad-just_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	$\geq 0$
min_iter	$sw_{iter}$	2	iteration number starting from which to enable switching	int
err_tol	$\epsilon_{tol}$	0.010	iteration error below which to enable switching	float
absViolation		1	use absolute (1) or relative (0) limit violation	(0, 1)

### 5.30.2 Slack

Slack generator.

## Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	$u$	connection status	1	<i>bool</i>	
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	<i>p.u.</i>	
q0	$q_0$	reactive power set point in system base	0	<i>p.u.</i>	
pmax	$p_{max}$	maximum active power in system base	999	<i>p.u.</i>	
pmin	$p_{min}$	minimum active power in system base	-1	<i>p.u.</i>	
qmax	$q_{max}$	maximum reactive power in system base	999	<i>p.u.</i>	
qmin	$q_{min}$	minimum reactive power in system base	-999	<i>p.u.</i>	
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		
a0	$\theta_0$	reference angle set point	0		
busv0	$V_{0bus}$		0		
busa0	$\theta_{0bus}$		0		

## Variables

Name	Symbol	Type	Description	Unit	Properties
q	$q$	Algeb	actual reactive power generation	<i>p.u.</i>	v_str
p	$p$	Algeb	actual active power generation	<i>p.u.</i>	v_str
a	$\theta$	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	$\theta$	ExtAlgeb	$\theta_0 u + \theta_{0bus} (1 - u)$
v	$V$	ExtAlgeb	$V_{0bus} (1 - u) + uv_0$

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
q	$q$	Algeb	$u \left( z_i^{qlim} (-V + v_0) + z_l^{qlim} (-q + q_{min}) + z_u^{qlim} (-q + q_{max}) \right)$
p	$p$	Algeb	$u \left( z_i^{plim} (-\theta + \theta_0) + z_l^{plim} (-p + p_{min}) + z_u^{plim} (-p + p_{max}) \right)$
a	$\theta$	ExtAlgeb	$-pu$
v	$V$	ExtAlgeb	$-qu$

## Discretes

Name	Symbol	Type	Info
qlim	$qlim$	SortedLimiter	
plim	$plim$	SortedLimiter	

Config Fields in [Slack]

Option	Symbol	Value	Info	Accepted values
al-low_adjust		1	allow adjusting upper or lower limits	(0, 1)
ad-just_lower		0	adjust lower limit	(0, 1)
ad-just_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	$\geq 0$
min_iter	$sw_{iter}$	2	iteration number starting from which to enable switching	int
err_tol	$\epsilon_{tol}$	0.010	iteration error below which to enable switching	float
absViolation		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)

## 5.31 StaticLoad

Static load group.

Common Parameters: u, name

Available models: *PQ*

### 5.31.1 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	<i>bool</i>	
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	<i>p.u.</i>	
q0	$q_0$	reactive power load in system base	0	<i>p.u.</i>	
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			

## Variables

Name	Symbol	Type	Description	Unit	Properties
a	$\theta$	ExtAlgeb			
v	$V$	ExtAlgeb			

## Initialization Equations

Name	Symbol	Type	Initial Value
a	$\theta$	ExtAlgeb	
v	$V$	ExtAlgeb	

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"	
a	$\theta$	ExtAlgebra	$u \left( I_{peq} V \gamma_{p2i} + P_{pf} \gamma_{p2p} + R_{eq} V^2 \gamma_{p2z} \right) \text{Indicator}(t_{dae} \geq 0) + u \left( R_{lb} V^2 z_l^{vcmp} + R_{ub} V^2 z_u^{vcmp} + p_0 z_i^{vcmp} \right) \text{Indicator}(t_{dae} < 0)$	+
v	$V$	ExtAlgebra	$u \left( I_{qeq} V \gamma_{q2i} + Q_{pf} \gamma_{q2q} + V^2 X_{eq} \gamma_{q2z} \right) \text{Indicator}(t_{dae} \geq 0) + u \left( V^2 X_{lb} z_l^{vcmp} + V^2 X_{ub} z_u^{vcmp} + q_0 z_i^{vcmp} \right) \text{Indicator}(t_{dae} < 0)$	+

## Services

Name	Symbol	Equation	Type
Rub	$R_{ub}$	$\frac{p_0}{v_{max}^2}$	ConstService
Xub	$X_{ub}$	$\frac{q_0}{v_{max}^2}$	ConstService
Rlb	$R_{lb}$	$\frac{p_0}{v_{min}^2}$	ConstService
Xlb	$X_{lb}$	$\frac{q_0}{v_{min}^2}$	ConstService
Ppf	$P_{pf}$	$R_{lb}V_0^2z_l^{vcmp} + R_{ub}V_0^2z_u^{vcmp} + p_0z_i^{vcmp}$	ConstService
Qpf	$Q_{pf}$	$V_0^2X_{lb}z_l^{vcmp} + V_0^2X_{ub}z_u^{vcmp} + q_0z_i^{vcmp}$	ConstService
Req	$R_{eq}$	$\frac{P_{pf}}{V_0^2}$	ConstService
Xeq	$X_{eq}$	$\frac{Q_{pf}}{V_0^2}$	ConstService
Ipeq	$I_{peq}$	$\frac{P_{pf}}{V_0}$	ConstService
Iqeq	$I_{qeq}$	$\frac{Q_{pf}}{V_0}$	ConstService

## Discretes

Name	Symbol	Type	Info
vcmp	$vcmp$	Limiter	

Config Fields in [PQ]

Option	Symbol	Value	Info	Accepted values
al-low_adjust		1	allow adjusting upper or lower limits	(0, 1)
ad-just_lower		0	adjust lower limit	(0, 1)
ad-just_upper		1	adjust upper limit	(0, 1)
pq2z	$z_{pq2z}$	1	pq2z conversion if out of voltage limits	(0, 1)
p2p	$\gamma_{p2p}$	0	P constant power percentage for TDS. Must have float (p2p+p2i+p2z)=1	float
p2i	$\gamma_{p2i}$	0	P constant current percentage	float
p2z	$\gamma_{p2z}$	1	P constant impedance percentage	float
q2q	$\gamma_{q2q}$	0	Q constant power percentage for TDS. Must have float (q2q+q2i+q2z)=1	float
q2i	$\gamma_{q2i}$	0	Q constant current percentage	float
q2z	$\gamma_{q2z}$	1	Q constant impedance percentage	float

## 5.32 StaticShunt

Static shunt compensator group.

Common Parameters: u, name

Available models: *Shunt*, *ShuntTD*, *ShuntSw*

### 5.32.1 Shunt

Phasor-domain shunt compensator Model.

#### Parameters

Name	Symbol	Description	Default	Unit	Properties
idx		unique device idx			
u	$u$	connection status	1	<i>bool</i>	
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		

#### Variables

Name	Symbol	Type	Description	Unit	Properties
a	$\theta$	ExtAlgeb			
v	$V$	ExtAlgeb			

#### Initialization Equations

Name	Symbol	Type	Initial Value
a	$\theta$	ExtAlgeb	
v	$V$	ExtAlgeb	

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
a	$\theta$	ExtAlgeb	$V^2 gu$
v	$V$	ExtAlgeb	$-V^2 bu$

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)

### 5.32.2 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		

#### Variables

Name	Symbol	Type	Description	Unit	Properties
vta	$V_{ta}$	Algeb			v_str
vtb	$V_{tb}$	Algeb			v_str
vtc	$V_{tc}$	Algeb			v_str
a	$\theta$	ExtAlgeb			
v	$V$	ExtAlgeb			

## Initialization Equations

Name	Symbol	Type	Initial Value
vta	$V_{ta}$	Algeb	$\frac{\sqrt{3}V \cos(\theta + 2\pi f_{systdae})}{3}$
vtb	$V_{tb}$	Algeb	$-\frac{\sqrt{3}V \cos(\theta + 2\pi f_{systdae} + \frac{\pi}{3})}{3}$
vtc	$V_{tc}$	Algeb	$-\frac{\sqrt{3}V \sin(\theta + 2\pi f_{systdae} + \frac{\pi}{6})}{3}$
a	$\theta$	ExtAlgeb	
v	$V$	ExtAlgeb	

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
vta	$V_{ta}$	Algeb	$\frac{\sqrt{3}V \cos(\theta + 2\pi f_{systdae})}{3} - V_{ta}$
vtb	$V_{tb}$	Algeb	$-\frac{\sqrt{3}V \cos(\theta + 2\pi f_{systdae} + \frac{\pi}{3})}{3} - V_{tb}$
vtc	$V_{tc}$	Algeb	$-\frac{\sqrt{3}V \sin(\theta + 2\pi f_{systdae} + \frac{\pi}{6})}{3} - V_{tc}$
a	$\theta$	ExtAlgeb	$V^2 gu$
v	$V$	ExtAlgeb	$-V^2 bu$

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)

### 5.32.3 ShuntSw

Switched Shunt Model.

Parameters  $gs$ ,  $bs$  and  $ns$  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	De-	Unit	Properties
			fault		
idx		unique device idx			
u	$u$	connection status	1	<i>bool</i>	
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	<i>p.u.</i>	y
bs		a list literal of switched susceptances blocks	0	<i>p.u.</i>	y
ns		a list literal of the element numbers in each switched block	[0]		
vref		voltage reference	1	<i>p.u.</i>	non_zero,non_negative
dv		voltage error deadband	0.050	<i>p.u.</i>	non_zero,non_negative
dt		delay before two consecutive switching	30	<i>sec- onds</i>	non_negative

## Variables

Name	Symbol	Type	Description	Unit	Properties
a	$\theta$	ExtAlgeb			
v	$V$	ExtAlgeb			

## Initialization Equations

Name	Symbol	Type	Initial Value
a	$\theta$	ExtAlgeb	
v	$V$	ExtAlgeb	

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
a	$\theta$	ExtAlgeb	$V^2geffu$
v	$V$	ExtAlgeb	$-V^2beffu$

## 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
al-low_adjust		1	allow adjusting upper or lower limits	(0, 1)
ad-just_lower		0	adjust lower limit	(0, 1)
ad-just_upper		1	adjust upper limit	(0, 1)
min_iter	$sw_{iter}$	2	iteration number starting from which to enable int switching	int
err_tol	$\epsilon_{tol}$	0.010	iteration error below which to enable switching	float

## 5.33 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`, `Sn`, `Vn`, `fn`, `bus`, `M`, `D`, `subidx`

Common Variables: `omega`, `delta`

Available models: *GENCLS*, *GENROU*, *PLBFVUI*

### 5.33.1 GENCLS

Classical generator model.

## Parameters

Name	Symbol	Description	De-fault	Unit	Properties
idx		unique device idx			
u	$u$	connection status	1	<i>bool</i>	
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	<i>MVA</i>	
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		<i>z</i>
xl	$x_l$	leakage reactance	0		<i>z</i>
xd1	$x'_d$	d-axis transient reactance	0.302		<i>z</i>
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	$\gamma_P$	P ratio of linked static gen	1		
gam-	$\gamma_Q$	Q ratio of linked static gen	1		
maq					
subidx		Generator idx in plant; only used by PSS/E data	0		

## Variables

Name	Symbol	Type	Description	Unit	Properties
delta	$\delta$	State	rotor angle	rad	v_str
omega	$\omega$	State	rotor speed	pu (Hz)	v_str
Id	$I_d$	Algeb	d-axis current		v_str
Iq	$I_q$	Algeb	q-axis current		v_str
vd	$V_d$	Algeb	d-axis voltage		v_str
vq	$V_q$	Algeb	q-axis voltage		v_str
tm	$\tau_m$	Algeb	mechanical torque		v_str
te	$\tau_e$	Algeb	electric torque		v_str
vf	$v_f$	Algeb	excitation voltage	pu	v_str
XadIfd	$X_{ad}I_{fd}$	Algeb	d-axis armature excitation current	p.u (kV)	v_str
Pe	$P_e$	Algeb	active power injection		v_str
Qe	$Q_e$	Algeb	reactive power injection		v_str
psid	$\psi_d$	Algeb	d-axis flux		v_str
psiq	$\psi_q$	Algeb	q-axis flux		v_str
a	$\theta$	ExtAlgeb	Bus voltage phase angle		
v	$V$	ExtAlgeb	Bus voltage magnitude		

## Initialization Equations

Name	Symbol	Type	Initial Value
delta	$\delta$	State	$\delta_0$
omega	$\omega$	State	$u$
Id	$I_d$	Algeb	$I_{d0}u$
Iq	$I_q$	Algeb	$I_{q0}u$
vd	$V_d$	Algeb	$V_{d0}u$
vq	$V_q$	Algeb	$V_{q0}u$
tm	$\tau_m$	Algeb	$\tau_{m0}$
te	$\tau_e$	Algeb	$\tau_{m0}u$
vf	$v_f$	Algeb	$uv_{f0}$
XadIfd	$X_{ad}I_{fd}$	Algeb	$uv_{f0}$
Pe	$P_e$	Algeb	$u(I_{d0}V_{d0} + I_{q0}V_{q0})$
Qe	$Q_e$	Algeb	$u(I_{d0}V_{q0} - I_{q0}V_{d0})$
psid	$\psi_d$	Algeb	$\psi_{d0}u$
psiq	$\psi_q$	Algeb	$\psi_{q0}u$
a	$\theta$	ExtAlgeb	
v	$V$	ExtAlgeb	

## Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
delta	$\delta$	State	$2\pi fu(\omega - 1)$	
omega	$\omega$	State	$u(-D(\omega - 1) - \tau_e + \tau_m)$	$M$

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
Id	$I_d$	Algeb	$I_d x'_d + \psi_d - v_f$
Iq	$I_q$	Algeb	$I_q x'_d + \psi_q$
vd	$V_d$	Algeb	$V u \sin(\delta - \theta) - V_d$
vq	$V_q$	Algeb	$V u \cos(\delta - \theta) - V_q$
tm	$\tau_m$	Algeb	$-\tau_m + \tau_{m0}$
te	$\tau_e$	Algeb	$-\tau_e + u(-I_d \psi_q + I_q \psi_d)$
vf	$v_f$	Algeb	$uv_{f0} - v_f$
XadIfd	$X_{ad} I_{fd}$	Algeb	$-X_{ad} I_{fd} + uv_{f0}$
Pe	$P_e$	Algeb	$-P_e + u(I_d V_d + I_q V_q)$
Qe	$Q_e$	Algeb	$-Q_e + u(I_d V_q - I_q V_d)$
psid	$\psi_d$	Algeb	$-\psi_d + u(I_q r_a + V_q)$
psiq	$\psi_q$	Algeb	$\psi_q + u(I_d r_a + V_d)$
a	$\theta$	ExtAlgeb	$-u(I_d V_d + I_q V_q)$
v	$V$	ExtAlgeb	$-u(I_d V_q - I_q V_d)$

## Services

Name	Symbol	Equation	Type
p0	$P_0$	$P_{0s}\gamma_P$	ConstService
q0	$Q_0$	$Q_{0s}\gamma_Q$	ConstService
_V	$V_c$	$Ve^{i\theta}$	ConstService
_S	$S$	$P_0 - iQ_0$	ConstService
_I	$I_c$	$\frac{S}{\text{conj}(V_c)}$	ConstService
_E	$E$	$I_c(r_a + ix'_d) + V_c$	ConstService
_deltac	$\delta_c$	$\log\left(\frac{E}{ E }\right)$	ConstService
delta0	$\delta_0$	$u \text{ im}(\delta_c)$	ConstService
vdq	$V_{dq}$	$V_cue^{-\delta_c+0.5i\pi}$	ConstService
Idq	$I_{dq}$	$I_cue^{-\delta_c+0.5i\pi}$	ConstService
Id0	$I_{d0}$	$\text{re}(I_{dq})$	ConstService
Iq0	$I_{q0}$	$\text{im}(I_{dq})$	ConstService
vd0	$V_{d0}$	$\text{re}(V_{dq})$	ConstService
vq0	$V_{q0}$	$\text{im}(V_{dq})$	ConstService
tm0	$\tau_{m0}$	$u(I_{d0}(I_{d0}r_a + V_{d0}) + I_{q0}(I_{q0}r_a + V_{q0}))$	ConstService
psid0	$\psi_{d0}$	$I_{q0}r_a u + V_{q0}$	ConstService
psiq0	$\psi_{q0}$	$-I_{d0}r_a u - V_{d0}$	ConstService
vf0	$v_{f0}$	$I_{d0}x'_d + I_{q0}r_a + V_{q0}$	ConstService

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	

### 5.33.2 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	<i>bool</i>	
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	$\gamma_P$	P ratio of linked static gen	1		
gammaq	$\gamma_Q$	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		
subidx		Generator idx in plant; only used by PSS/E data	0		

## Variables

Name	Symbol	Type	Description	Unit	Properties
delta	$\delta$	State	rotor angle	rad	v_str
omega	$\omega$	State	rotor speed	pu (Hz)	v_str
e1q	$e'_q$	State	q-axis transient voltage		v_str
e1d	$e'_d$	State	d-axis transient voltage		v_str
e2d	$e''_d$	State	d-axis sub-transient voltage		v_str
e2q	$e''_q$	State	q-axis sub-transient voltage		v_str
Id	$I_d$	Algeb	d-axis current		v_str
Iq	$I_q$	Algeb	q-axis current		v_str
vd	$V_d$	Algeb	d-axis voltage		v_str
vq	$V_q$	Algeb	q-axis voltage		v_str
tm	$\tau_m$	Algeb	mechanical torque		v_str
te	$\tau_e$	Algeb	electric torque		v_str
vf	$v_f$	Algeb	excitation voltage	pu	v_str
XadIfd	$X_{ad}I_{fd}$	Algeb	d-axis armature excitation current	p.u (kV)	v_str
Pe	$P_e$	Algeb	active power injection		v_str
Qe	$Q_e$	Algeb	reactive power injection		v_str
psid	$\psi_d$	Algeb	d-axis flux		v_str
psiq	$\psi_q$	Algeb	q-axis flux		v_str
psi2q	$\psi_{aq}$	Algeb	q-axis air gap flux		v_str
psi2d	$\psi_{ad}$	Algeb	d-axis air gap flux		v_str
psi2	$\psi_a$	Algeb	air gap flux magnitude		v_str
Se	$S_e( \psi_a )$	Algeb	saturation output		v_str
XaqI1q	$X_{aq}I_{1q}$	Algeb	q-axis reaction	p.u (kV)	v_str
a	$\theta$	ExtAlgeb	Bus voltage phase angle		
v	$V$	ExtAlgeb	Bus voltage magnitude		

## Initialization Equations

Name	Symbol	Type	Initial Value
delta	$\delta$	State	$\delta_0$
omega	$\omega$	State	$u$
e1q	$e'_q$	State	$ue'_{q0}$
e1d	$e'_d$	State	$ue'_{d0}$
e2d	$e''_d$	State	$ue''_{d0}$
e2q	$e''_q$	State	$ue''_{q0}$
Id	$I_d$	Algeb	$I_{d0}u$
Iq	$I_q$	Algeb	$I_{q0}u$
vd	$V_d$	Algeb	$V_{d0}u$
vq	$V_q$	Algeb	$V_{q0}u$
tm	$\tau_m$	Algeb	$\tau_{m0}$
te	$\tau_e$	Algeb	$\tau_{m0}u$
vf	$v_f$	Algeb	$uv_{f0}$
XadIfd	$X_{ad}I_{fd}$	Algeb	$uv_{f0}$
Pe	$P_e$	Algeb	$u(I_{d0}V_{d0} + I_{q0}V_{q0})$
Qe	$Q_e$	Algeb	$u(I_{d0}V_{q0} - I_{q0}V_{d0})$
psid	$\psi_d$	Algeb	$\psi_{d0}u$
psiq	$\psi_q$	Algeb	$\psi_{q0}u$
psi2q	$\psi_{aq}$	Algeb	$\psi_{aq0}u$
psi2d	$\psi_{ad}$	Algeb	$\psi_{ad0}u$
psi2	$\psi_a$	Algeb	$u \left  \psi''_{0,dq} \right $
Se	$S_e( \psi_a )$	Algeb	$S_{e0}u$
XaqI1q	$X_{aq}I_{1q}$	Algeb	0
a	$\theta$	ExtAlgeb	
v	$V$	ExtAlgeb	

## Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
delta	$\delta$	State	$2\pi fu(\omega - 1)$	
omega	$\omega$	State	$u(-D(\omega - 1) - \tau_e + \tau_m)$	$M$
e1q	$e'_q$	State	$-X_{ad}I_{fd} + v_f$	$T'_{d0}$
e1d	$e'_d$	State	$-X_{aq}I_{1q}$	$T'_{q0}$
e2d	$e''_d$	State	$-I_d(x'_d - x_l) - e''_d + e'_q$	$T''_{d0}$
e2q	$e''_q$	State	$I_q(-x_l + x'_q) - e''_q + e'_d$	$T''_{q0}$

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
Id	$I_d$	Algeb	$I_d x_d'' + \psi_d - \psi_{ad}$
Iq	$I_q$	Algeb	$I_q x_q'' + \psi_q + \psi_{aq}$
vd	$V_d$	Algeb	$V u \sin(\delta - \theta) - V_d$
vq	$V_q$	Algeb	$V u \cos(\delta - \theta) - V_q$
tm	$\tau_m$	Algeb	$-\tau_m + \tau_{m0}$
te	$\tau_e$	Algeb	$-\tau_e + u(-I_d \psi_q + I_q \psi_d)$
vf	$v_f$	Algeb	$uv f_0 - v_f$
XadIfd	$X_{ad} I_{fd}$	Algeb	$-X_{ad} I_{fd} + u(S_e( \psi_a )\psi_{ad} + e'_q + (-x'_d + x_d)(I_d \gamma_{d1} - \gamma_{d2} e''_d + \gamma_{d2} e'_q))$
Pe	$P_e$	Algeb	$-P_e + u(I_d V_d + I_q V_q)$
Qe	$Q_e$	Algeb	$-Q_e + u(I_d V_q - I_q V_d)$
psid	$\psi_d$	Algeb	$-\psi_d + u(I_q r_a + V_q)$
psiq	$\psi_q$	Algeb	$\psi_q + u(I_d r_a + V_d)$
psi2q	$\psi_{aq}$	Algeb	$\gamma_{q1} e'_d - \psi_{aq} + e''_q (1 - \gamma_{q1})$
psi2d	$\psi_{ad}$	Algeb	$\gamma_{d1} e'_q + \gamma_{d2} e''_d (x'_d - x_l) - \psi_{ad}$
psi2	$\psi_a$	Algeb	$-\psi_a^2 + \psi_{ad}^2 + \psi_{aq}^2$
Se	$S_e( \psi_a )$	Algeb	$B_{S_{AT}}^q z_0^{SL} \left( -A_{S_{AT}}^q + \psi_a \right)^2 - S_e( \psi_a ) \psi_a$
XaqI1q	$X_{aq} I_{1q}$	Algeb	$S_e( \psi_a ) \gamma_{qd} \psi_{aq} - X_{aq} I_{1q} + e'_d + (x_q - x'_q)(-I_q \gamma_{q1} - \gamma_{q2} e''_q + \gamma_{q2} e'_d)$
a	$\theta$	ExtAl- geb	$-u(I_d V_d + I_q V_q)$
v	$V$	ExtAl- geb	$-u(I_d V_q - I_q V_d)$

## Services

Name	Symbol	Equation	Type
p0	$P_0$	$P_{0s} \gamma_P$	ConstService
q0	$Q_0$	$Q_{0s} \gamma_Q$	ConstService
gd1	$\gamma_{d1}$	$\frac{-x_l + x_d''}{x_d' - x_l}$	ConstService
gq1	$\gamma_{q1}$	$\frac{-x_l + x_q''}{x_j' - x_q}$	ConstService
gd2	$\gamma_{d2}$	$\frac{x_q' - x_d'}{(x_d' - x_l)^2}$	ConstService
gq2	$\gamma_{q2}$	$\frac{-x_q'' + x_q'}{(-x_l + x_q')^2}$	ConstService
gqd	$\gamma_{qd}$	$\frac{-x_l + x_q}{x_d - x_l}$	ConstService
_S12	$S_{1.2}$	$S_{1.2} - f S12 + 1$	ConstService
SAT_E1	$E_{S_{AT}}^{1c}$	1.0	ConstService
SAT_E2	$E_{S_{AT}}^{2c}$	$3.2 - 2z_{S_{AT}}^{SE2}$	ConstService
SAT_SE1	$SE_{S_{AT}}^{1c}$	$S_{1.0}$	ConstService

continues on next page

Table 45 – continued from previous page

Name	Symbol	Equation	Type
SAT_SE2	$SE_{SAT}^{2c}$	$S_{1,2} - 2z_{SAT}^{SE2} + 2$	ConstService
SAT_a	$a_{SAT}$	$\sqrt{\frac{E_{SAT}^{1c}}{E_{SAT}^{2c}}} \left( \text{Indicator}(SE_{SAT}^{2c} > 0) + \text{Indicator}(SE_{SAT}^{2c} < 0) \right)$	ConstService
SAT_A	$A_{SAT}^q$	$E_{SAT}^{2c} - \frac{E_{SAT}^{1c} - E_{SAT}^{2c}}{a_{SAT} - 1}$	ConstService
SAT_B	$B_{SAT}^q$	$\frac{E_{SAT}^{2c} SE_{SAT}^{2c} (a_{SAT} - 1)^2 (\text{Indicator}(a_{SAT} > 0) + \text{Indicator}(a_{SAT} < 0))}{(E_{SAT}^{1c} - E_{SAT}^{2c})^2}$	ConstService
_V	$V_c$	$Ve^{i\theta}$	ConstService
_S	$S$	$P_0 - iQ_0$	ConstService
_Zs	$Z_s$	$r_a + ix_d''$	ConstService
_It	$I_t$	$\frac{S}{\text{conj}(V_c)}$	ConstService
_Is	$I_s$	$I_t + \frac{V_c}{Z_s}$	ConstService
psi20	$\psi_0''$	$I_s Z_s$	ConstService
psi20_arg	$\theta_{\psi''0}$	$\arg(\psi_0'')$	ConstService
psi20_abs	$ \psi_0'' $	$ \psi_0'' $	ConstService
_It_arg	$\theta_{It0}$	$\arg(I_t)$	ConstService
_psi20_It_arg	$\theta_{\psi a It}$	$-\theta_{It0} + \theta_{\psi''0}$	ConstService
Se0	$S_{e0}$	$\frac{B_{SAT}^q (-A_{SAT}^q +  \psi_0'' )^2 \text{Indicator}( \psi_0''  \geq A_{SAT}^q)}{ \psi_0'' }$	ConstService
_a	$a'$	$ \psi_0''  (S_{e0} \gamma_{qd} + 1)$	ConstService
_b	$b'$	$(-x_q + x_q'')  I_t $	ConstService
delta0	$\delta_0$	$\theta_{\psi''0} + \text{atan}\left(\frac{b' \cos(\theta_{\psi a It})}{-a' + b' \sin(\theta_{\psi a It})}\right)$	ConstService
_Tdq	$T_{dq}$	$-i \sin(\delta_0) + \cos(\delta_0)$	ConstService
psi20_dq	$\psi_{0,dq}''$	$T_{dq} \psi_0''$	ConstService
It_dq	$I_{t,dq}$	$\text{conj}(I_t T_{dq})$	ConstService
psi2d0	$\psi_{ad0}$	$\text{re}(\psi_{0,dq}'')$	ConstService
psi2q0	$\psi_{aq0}$	$-\text{im}(\psi_{0,dq}'')$	ConstService
Id0	$I_{d0}$	$\text{im}(I_{t,dq})$	ConstService
Iq0	$I_{q0}$	$\text{re}(I_{t,dq})$	ConstService
vd0	$V_{d0}$	$-I_{d0} r_a + I_{q0} x_q'' + \psi_{aq0}$	ConstService
vq0	$V_{q0}$	$-I_{d0} x_d'' - I_{q0} r_a + \psi_{ad0}$	ConstService
tm0	$\tau_{m0}$	$u(I_{d0}(I_{d0} r_a + V_{d0}) + I_{q0}(I_{q0} r_a + V_{q0}))$	ConstService
vf0	$v_{f0}$	$I_{d0}(x_d - x_d'') + \psi_{ad0}(S_{e0} + 1)$	ConstService
psid0	$\psi_{d0}$	$I_{q0} r_a u + V_{q0}$	ConstService
psiq0	$\psi_{q0}$	$-I_{d0} r_a u - V_{d0}$	ConstService
e1q0	$e'_{q0}$	$I_{d0}(x_d' - x_d) - S_{e0} \psi_{ad0} + v_{f0}$	ConstService
e1d0	$e'_{d0}$	$I_{q0}(x_q - x_q') - S_{e0} \gamma_{qd} \psi_{aq0}$	ConstService
e2d0	$e''_{d0}$	$I_{d0}(-x_d + x_l) - S_{e0} \psi_{ad0} + v_{f0}$	ConstService
e2q0	$e''_{q0}$	$-I_{q0}(x_l - x_q) - S_{e0} \gamma_{qd} \psi_{aq0}$	ConstService

## Discretes

Name	Symbol	Type	Info
SL	$SL$	LessThan	

## Blocks

Name	Symbol	Type	Info
SAT	$S_{AT}$	ExcQuadSat	

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	

### 5.33.3 PLVFU1

PLVFU1 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_plbfu1.xlsx` and `plbfu.xlsx` in `andes/cases/ieee14` for an example.

Voltage and frequency 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
subidx		Generator idx in plant; only used by PSS/E data	0		

## Variables

Name	Symbol	Type	Description	Unit	Properties
Vflt	$V_{flt}$	State	filtered voltage	pu	v_str
omega	$\omega$	State	filtered frequency	pu	v_str
delta	$\delta$	State	rotor angle	rad	v_str
a	$\theta$	ExtAlgeb	Bus voltage phase angle		
v	$V$	ExtAlgeb	Bus voltage magnitude		

## Initialization Equations

Name	Symbol	Type	Initial Value
Vflt	$V_{flt}$	State	$1/V_{scale}V_{ts} - V_{offs}$
omega	$\omega$	State	$1/f_{scale}f_{ts} - f_{offs}$
delta	$\delta$	State	$\delta_0$
a	$\theta$	ExtAlgeb	
v	$V$	ExtAlgeb	

## Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
Vflt	$V_{flt}$	State	$1/V_{scale}Vts - V_{flt} - V_{offs}$	$T_v$
omega	$\omega$	State	$1/f_{scale}fts - \omega - f_{offs}$	$T_f$
delta	$\delta$	State	$2\pi f_n u (\omega - 1)$	

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
a	$\theta$	ExtAlgeb	$-\frac{VV_{flt}x_s \sin(\delta-\theta)}{r_a^2+x_s^2} + \frac{Vr_a(V-V_{flt} \cos(\delta-\theta))}{r_a^2+x_s^2}$
v	$V$	ExtAlgeb	$\frac{VV_{flt}r_a \sin(\delta-\theta)}{r_a^2+x_s^2} + \frac{Vx_s(V-V_{flt} \cos(\delta-\theta))}{r_a^2+x_s^2}$

## Services

Name	Symbol	Equation	Type
zs	$zs$	$r_a + ix_s$	ConstService
zs2n	$zs2n$	$r_a^2 - x_s^2$	ConstService
Ec	$E_c$	$Ve^{i\theta} + (r_a + ix_s) \operatorname{conj}\left(\frac{(p+iq)e^{-i\theta}}{V}\right)$	ConstService
E0	$E_0$	$ E_c $	ConstService
delta0	$\delta_0$	$\arg(E_c)$	ConstService
Vts	$Vts$	0	ConstService
fts	$fts$	0	ConstService
ifscale	$1/f_{scale}$	$\frac{1}{f_{scale}}$	ConstService
iVscale	$1/V_{scale}$	$\frac{1}{V_{scale}}$	ConstService
foffs	$f_{offs}$	$1/f_{scale}fts - 1$	ConstService
Voffs	$V_{offs}$	$1/V_{scale}Vts - E_0$	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)

## 5.34 TimedEvent

Timed event group

Common Parameters: u, name

Available models: *Toggle*, *Fault*, *Alter*

### 5.34.1 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	<i>u</i>	connection status	1		<i>bool</i>
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

#### Services

Name	Symbol	Equation	Type
_u	<i>u</i>	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)

## 5.34.2 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,  $t_f$  and  $t_c$ , for the fault-on and fault-clearance time, respectively, although only  $t_f$  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  $x_f$  and  $r_f$ .

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  $x_f$ .

## 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)	

## Variables

Name	Symbol	Type	Description	Unit	Properties
a	$\theta$	ExtAlgeb	Bus voltage angle	p.u.(kV)	
v	$V$	ExtAlgeb	Bus voltage magnitude	p.u.(kV)	

## Initialization Equations

Name	Symbol	Type	Initial Value
a	$\theta$	ExtAlgeb	
v	$V$	ExtAlgeb	

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
a	$\theta$	ExtAlgeb	$V^2 g_f u u_f$
v	$V$	ExtAlgeb	$-V^2 b_f u u_f$

## Services

Name	Symbol	Equation	Type
gf	$g_f$	$\frac{\operatorname{re}(x_f) - \operatorname{im}(x_f)}{(\operatorname{re}(x_f) - \operatorname{im}(x_f))^2 + (\operatorname{re}(x_f) + \operatorname{im}(x_f))^2}$	ConstService
bf	$b_f$	$\frac{-\operatorname{re}(x_f) - \operatorname{im}(x_f)}{(\operatorname{re}(x_f) - \operatorname{im}(x_f))^2 + (\operatorname{re}(x_f) + \operatorname{im}(x_f))^2}$	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	

### 5.34.3 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  $P_{pf}$  and  $Q_{pf}$ . Altering  $p_0$  and  $q_0$  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	<i>bool</i>	
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 random sampling	0		
lb		lower bound of random sampling	0		
ub		upper bound of random sampling	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)

## 5.35 TurbineGov

Turbine governor group for synchronous generator.

Common Parameters: u, name

Common Variables: pout

Available models: *TG2*, *TGOV1*, *TGOVIDB*, *TGOVIN*, *TGOVINDB*, *IEEEG1*, *IEESGO*, *GAST*, *HYGOV*, *HYGOVDB*, *HYGOV4*

### 5.35.1 TG2

#### Parameters

Name	Symbol	Description	De-fault	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 $S_n$ if not provided.	0	MVA	
wref0	$\omega_{ref0}$	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		
Sg	$S_n$	Rated power from generator	0	MVA	
ug	$u_g$	Generator connection status	0	bool	
Vn	$V_n$	Rated voltage from generator	0	kV	

## Variables

Name	Symbol	Type	Description	Unit	Properties
ll_x	$x'_{ll}$	State	State in lead-lag	v_str	
omega	$\omega$	ExtState	Generator speed	p.u.	
paux	$P_{aux}$	Algeb	Auxiliary power input	v_str	
pout	$P_{out}$	Algeb	Turbine final output power	v_str	
wref	$\omega_{ref}$	Algeb	Speed reference variable	v_str	
w_d	$\omega_{dev}$	Algeb	Generator speed deviation before dead band (positive for under speed)	v_str	
w_dm	$\omega_{dm}$	Algeb	Measured speed deviation after dead band	v_str	
w_dmg	$\omega_{dmG}$	Algeb	Speed deviation after dead band after gain	v_str	
ll_y	$y_{ll}$	Algeb	Output of lead-lag	v_str	
pnl	$P_{nl}$	Algeb	Power output before hard limiter	v_str	
tm	$\tau_m$	ExtAlgeb	Mechanical power interface to SynGen		

## Initialization Equations

Name	Symbol	Type	Initial Value
ll_x	$x'_{ll}$	State	$\omega_{dmG}$
omega	$\omega$	ExtState	
paux	$P_{aux}$	Algeb	$P_{aux0}$
pout	$P_{out}$	Algeb	$\tau_{m0} u_e$
wref	$\omega_{ref}$	Algeb	$\omega_{ref0}$
w_d	$\omega_{dev}$	Algeb	0
w_dm	$\omega_{dm}$	Algeb	0
w_dmg	$\omega_{dmG}$	Algeb	0
ll_y	$y_{ll}$	Algeb	$\omega_{dmG}$
pnl	$P_{nl}$	Algeb	$\tau_{m0}$
tm	$\tau_m$	ExtAlgeb	

## Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
ll_x	$x'_{ll}$	State	$\omega_{dmG} - x'_{ll}$	$T_2$
omega	$\omega$	ExtState	0	

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
paux	$P_{aux}$	Algeb	$P_{aux0} - P_{aux}$
pout	$P_{out}$	Algeb	$P_{nl}z_i^{plim} - P_{out} + p_{max}z_u^{plim} + p_{min}z_l^{plim}$
wref	$\omega_{ref}$	Algeb	$\omega_{ref0} - \omega_{ref}$
w_d	$\omega_{dev}$	Algeb	$-\omega_{dev} + u_e (-\omega + \omega_{ref})$
w_dm	$\omega_{dm}$	Algeb	$L_{db}z_l r^{w_db} + U_{db}z_u r^{w_db} + \omega_{dev} (1 - z_i^{w_db}) - \omega_{dm}$
w_dmg	$\omega_{dmG}$	Algeb	$G\omega_{dm} - \omega_{dmG}$
ll_y	$y_{ll}$	Algeb	$T_1 (\omega_{dmG} - x'_{ll}) + T_2 x'_{ll} - T_2 y_{ll} + (z_1^{LT_{ll}})^2 (-x'_{ll} + y_{ll})$
pnl	$P_{nl}$	Algeb	$-P_{nl} + P_{ref0} + y_{ll}$
tm	$\tau_m$	ExtAlgeb	$u_e (P_{out} - \tau_{m0})$

## Services

Name	Symbol	Equation	Type
ue	$u_e$	$uu_g$	ConstService
pref0	$P_{ref0}$	$\tau_{m0}$	ConstService
paux0	$P_{aux0}$	0	ConstService
gain	$G$	$\frac{u}{R}$	ConstService

## Discretes

Name	Symbol	Type	Info
w_db	$w_db$	DeadBandRT	
ll_LT1	$LT_{ll}$	LessThan	
ll_LT2	$LT_{ll}$	LessThan	
plim	$plim$	HardLimiter	

## Blocks

Name	Symbol	Type	Info
ll	$ll$	LeadLag	

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)

### 5.35.2 TGOV1

TGOV1 turbine governor model.

Implements the PSS/E TGOV1 model without deadband.

#### Parameters

Name	Sym- bol	Description	De- fault	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 $S_n$ if not provided.	0	MVA	
wref0	$\omega_{ref0}$	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
Sg	$S_n$	Rated power from generator	0	MVA	
ug	$u_g$	Generator connection status	0	bool	
Vn	$V_n$	Rated voltage from generator	0	kV	

## Variables

Name	Symbol	Type	Description	Unit	Properties
LAG_y	$y_{LAG}$	State	State in lag TF		v_str
LL_x	$x'_{LL}$	State	State in lead-lag		v_str
omega	$\omega$	ExtState	Generator speed	p.u.	
paux	$P_{aux}$	Algeb	Auxiliary power input		v_str
pout	$P_{out}$	Algeb	Turbine final output power		v_str
wref	$\omega_{ref}$	Algeb	Speed reference variable		v_str
pref	$P_{ref}$	Algeb	Reference power input		v_str
wd	$\omega_{dev}$	Algeb	Generator speed deviation	p.u.	v_str
pd	$P_d$	Algeb	Pref plus speed deviation times gain	p.u.	v_str
LL_y	$y_{LL}$	Algeb	Output of lead-lag		v_str
tm	$\tau_m$	ExtAlgeb	Mechanical power interface to SynGen		

## Initialization Equations

Name	Symbol	Type	Initial Value
LAG_y	$y_{LAG}$	State	$P_d$
LL_x	$x'_{LL}$	State	$y_{LAG}$
omega	$\omega$	ExtState	
paux	$P_{aux}$	Algeb	$P_{aux0}$
pout	$P_{out}$	Algeb	$\tau_m u_e$
wref	$\omega_{ref}$	Algeb	$\omega_{ref0}$
pref	$P_{ref}$	Algeb	$R\tau_m$
wd	$\omega_{dev}$	Algeb	0
pd	$P_d$	Algeb	$\tau_m u_e$
LL_y	$y_{LL}$	Algeb	$y_{LAG}$
tm	$\tau_m$	ExtAlgeb	

## Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
LAG_y	$y_{LAG}$	State	$P_d - y_{LAG}$	$T_1$
LL_x	$x'_{LL}$	State	$-x'_{LL} + y_{LAG}$	$T_3$
omega	$\omega$	ExtState	0	

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
paux	$P_{aux}$	Algeb	$P_{aux0} - P_{aux}$
pout	$P_{out}$	Algeb	$-P_{out} + u_e (-D_t \omega_{dev} + y_{LL})$
wref	$\omega_{ref}$	Algeb	$\omega_{ref0} - \omega_{ref}$
pref	$P_{ref}$	Algeb	$P_{ref0}R - P_{ref}$
wd	$\omega_{dev}$	Algeb	$-\omega_{dev} + u_e (\omega - \omega_{ref})$
pd	$P_d$	Algeb	$G u_e (P_{aux} + P_{ref} - \omega_{dev}) - P_d$
LL_y	$y_{LL}$	Algeb	$T_2 (-x'_{LL} + y_{LAG}) + T_3 x'_{LL} - T_3 y_{LL} + \left( z_1^{LT_{LL}} \right)^2 (-x'_{LL} + y_{LL})$
tm	$\tau_m$	ExtAlgeb	$u_e (P_{out} - \tau_{m0})$

## Services

Name	Symbol	Equation	Type
ue	$u_e$	$uu_g$	ConstService
pref0	$P_{ref0}$	$\tau_{m0}$	ConstService
paux0	$P_{aux0}$	0	ConstService
gain	$G$	$\frac{u_e}{R}$	ConstService

## Discretes

Name	Symbol	Type	Info
LAG_lim	$lim_{LAG}$	AntiWindup	Limiter in Lag
LL_LT1	$LT_{LL}$	LessThan	
LL_LT2	$LT_{LL}$	LessThan	

## Blocks

Name	Symbol	Type	Info
LAG	$LAG$	LagAntiWindup	
LL	$LL$	LeadLag	

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)

### 5.35.3 TGOV1DB

TGOV1 turbine governor model with speed input deadband.

#### Parameters

Name	Symbol	Description	De-fault	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 $S_n$ if not provided.	0	MVA	
wref0	$\omega_{ref0}$	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.	
Sg	$S_n$	Rated power from generator	0	MVA	
ug	$u_g$	Generator connection status	0	bool	
Vn	$V_n$	Rated voltage from generator	0	kV	

## Variables

Name	Symbol	Type	Description	Unit	Properties
LAG_y	$y_{LAG}$	State	State in lag TF		v_str
LL_x	$x'_{LL}$	State	State in lead-lag		v_str
omega	$\omega$	ExtState	Generator speed	p.u.	
paux	$P_{aux}$	Algeb	Auxiliary power input		v_str
pout	$P_{out}$	Algeb	Turbine final output power		v_str
wref	$\omega_{ref}$	Algeb	Speed reference variable		v_str
pref	$P_{ref}$	Algeb	Reference power input		v_str
wd	$\omega_{dev}$	Algeb	Generator speed deviation	p.u.	v_str
pd	$P_d$	Algeb	Pref plus speed deviation times gain	p.u.	v_str
LL_y	$y_{LL}$	Algeb	Output of lead-lag		v_str
DB_y	$y_{DB}$	Algeb	Deadband type 1 output		v_str
tm	$\tau_m$	ExtAlgeb	Mechanical power interface to SynGen		

## Initialization Equations

Name	Symbol	Type	Initial Value
LAG_y	$y_{LAG}$	State	$P_d$
LL_x	$x'_{LL}$	State	$y_{LAG}$
omega	$\omega$	ExtState	
paux	$P_{aux}$	Algeb	$P_{aux0}$
pout	$P_{out}$	Algeb	$\tau_m u_e$
wref	$\omega_{ref}$	Algeb	$\omega_{ref0}$
pref	$P_{ref}$	Algeb	$R\tau_m$
wd	$\omega_{dev}$	Algeb	0
pd	$P_d$	Algeb	$\tau_m u_e$
LL_y	$y_{LL}$	Algeb	$y_{LAG}$
DB_y	$y_{DB}$	Algeb	$1.0z_l^{db_{DB}}(\omega_{dev} - db_L) + 1.0z_u^{db_{DB}}(\omega_{dev} - db_U)$
tm	$\tau_m$	ExtAlgeb	

## Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
LAG_y	$y_{LAG}$	State	$P_d - y_{LAG}$	$T_1$
LL_x	$x'_{LL}$	State	$-x'_{LL} + y_{LAG}$	$T_3$
omega	$\omega$	ExtState	0	

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
paux	$P_{aux}$	Algeb	$P_{aux0} - P_{aux}$
pout	$P_{out}$	Algeb	$-D_t y_{DB} - P_{out} + y_{LL}$
wref	$\omega_{ref}$	Algeb	$\omega_{ref0} - \omega_{ref}$
pref	$P_{ref}$	Algeb	$P_{ref0}R - P_{ref}$
wd	$\omega_{dev}$	Algeb	$-\omega_{dev} + u_e (\omega - \omega_{ref})$
pd	$P_d$	Algeb	$G u_e (P_{aux} + P_{ref} - y_{DB}) - P_d$
LL_y	$y_{LL}$	Algeb	$T_2 (-x'_{LL} + y_{LAG}) + T_3 x'_{LL} - T_3 y_{LL} + (z_1^{LT_{LL}})^2 (-x'_{LL} + y_{LL})$
DB_y	$y_{DB}$	Algeb	$-y_{DB} + 1.0 z_l^{db_{DB}} (\omega_{dev} - db_L) + 1.0 z_u^{db_{DB}} (\omega_{dev} - db_U)$
tm	$\tau_m$	ExtAlgeb	$u_e (P_{out} - \tau_{m0})$

## Services

Name	Symbol	Equation	Type
ue	$u_e$	$uu_g$	ConstService
pref0	$P_{ref0}$	$\tau_{m0}$	ConstService
paux0	$P_{aux0}$	0	ConstService
gain	$G$	$\frac{u_e}{R}$	ConstService

## Discretes

Name	Symbol	Type	Info
LAG_lim	$lim_{LAG}$	AntiWindup	Limiter in Lag
LL_LT1	$LT_{LL}$	LessThan	
LL_LT2	$LT_{LL}$	LessThan	
DB_db	$db_{DB}$	DeadBand	

## Blocks

Name	Symbol	Type	Info
LAG	$LAG$	LagAntiWindup	
LL	$LL$	LeadLag	
DB	$DB$	DeadBand1	deadband for speed deviation

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)

### 5.35.4 TGOV1N

New TGOV1 (TGOV1N) turbine governor model.

The TGOV1N model that sums `pref` and `paux` signals after the droop. This model is useful for incorporating AGC and scheduling signals, which will not be multiplied by  $1/R$  like in the original TGOV1 model.

Scheduling changes should write to `pref0.v` in place. AGC signal should write to `paux0.v` in place.

Modifying `tmo` is not allowed.

#### Examples

To update all `paux0` values to `paux_new`, which contains the new values, do

```
ss.TGOV1N.paux0.v[:] = paux_new # in-place update of the `paux0.v` array
```

instead of

```
ss.TGOV1N.paux0.v = paux_new # error; changes the reference of `paux0.v`
```

## Parameters

Name	Symbol	Description	De-fault	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 $S_n$ if not provided.	0	MVA	
wref0	$\omega_{ref0}$	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
Sg	$S_n$	Rated power from generator	0	MVA	
ug	$u_g$	Generator connection status	0	bool	
Vn	$V_n$	Rated voltage from generator	0	kV	

## Variables

Name	Symbol	Type	Description	Unit	Properties
LAG_y	$y_{LAG}$	State	State in lag TF		v_str
LL_x	$x'_{LL}$	State	State in lead-lag		v_str
omega	$\omega$	ExtState	Generator speed	p.u.	
paux	$P_{aux}$	Algeb	Auxiliary power input		v_str
pout	$P_{out}$	Algeb	Turbine final output power		v_str
wref	$\omega_{ref}$	Algeb	Speed reference variable		v_str
pref	$P_{ref}$	Algeb	Reference power input		v_str
wd	$\omega_{dev}$	Algeb	Generator speed deviation	p.u.	v_str
pd	$P_d$	Algeb	Pref plus speed deviation times gain	p.u.	v_str
LL_y	$y_{LL}$	Algeb	Output of lead-lag		v_str
tm	$\tau_m$	ExtAlgeb	Mechanical power interface to SynGen		

## Initialization Equations

Name	Symbol	Type	Initial Value
LAG_y	$y_{LAG}$	State	$P_d$
LL_x	$x'_{LL}$	State	$y_{LAG}$
omega	$\omega$	ExtState	
paux	$P_{aux}$	Algeb	$P_{aux0}$
pout	$P_{out}$	Algeb	$\tau_{m0} u_e$
wref	$\omega_{ref}$	Algeb	$\omega_{ref0}$
pref	$P_{ref}$	Algeb	$\tau_{m0}$
wd	$\omega_{dev}$	Algeb	0
pd	$P_d$	Algeb	$\tau_{m0} u_e$
LL_y	$y_{LL}$	Algeb	$y_{LAG}$
tm	$\tau_m$	ExtAlgeb	

## Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
LAG_y	$y_{LAG}$	State	$P_d - y_{LAG}$	$T_1$
LL_x	$x'_{LL}$	State	$-x'_{LL} + y_{LAG}$	$T_3$
omega	$\omega$	ExtState	0	

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
paux	$P_{aux}$	Algeb	$P_{aux0} - P_{aux}$
pout	$P_{out}$	Algeb	$-P_{out} + u_e (-D_t \omega_{dev} + y_{LL})$
wref	$\omega_{ref}$	Algeb	$\omega_{ref0} - \omega_{ref}$
pref	$P_{ref}$	Algeb	$P_{ref0} - P_{ref}$
wd	$\omega_{dev}$	Algeb	$-\omega_{dev} + u_e (\omega - \omega_{ref})$
pd	$P_d$	Algeb	$-P_d + u_e (-G\omega_{dev} + P_{aux} + P_{ref})$
LL_y	$y_{LL}$	Algeb	$T_2 (-x'_{LL} + y_{LAG}) + T_3 x'_{LL} - T_3 y_{LL} + (z_1^{LT_{LL}})^2 (-x'_{LL} + y_{LL})$
tm	$\tau_m$	ExtAlgeb	$u_e (P_{out} - \tau_{m0})$

## Services

Name	Symbol	Equation	Type
ue	$u_e$	$uu_g$	ConstService
pref0	$P_{ref0}$	$\tau_{m0}$	ConstService
paux0	$P_{aux0}$	0	ConstService
gain	$G$	$\frac{u_e}{R}$	ConstService

## Discretes

Name	Symbol	Type	Info
LAG_lim	$lim_{LAG}$	AntiWindup	Limiter in Lag
LL_LT1	$LT_{LL}$	LessThan	
LL_LT2	$LT_{LL}$	LessThan	

## Blocks

Name	Symbol	Type	Info
LAG	$LAG$	LagAntiWindup	
LL	$LL$	LeadLag	

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)

### 5.35.5 TGOV1NDB

TGOV1N turbine governor model with speed input deadband.

## Parameters

Name	Symbol	Description	De-fault	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 $S_n$ if not provided.	0	MVA	
wref0	$\omega_{ref0}$	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.	
Sg	$S_n$	Rated power from generator	0	MVA	
ug	$u_g$	Generator connection status	0	bool	
Vn	$V_n$	Rated voltage from generator	0	kV	

## Variables

Name	Symbol	Type	Description	Unit	Properties
LAG_y	$y_{LAG}$	State	State in lag TF		v_str
LL_x	$x'_{LL}$	State	State in lead-lag		v_str
omega	$\omega$	ExtState	Generator speed	p.u.	
paux	$P_{aux}$	Algeb	Auxiliary power input		v_str
pout	$P_{out}$	Algeb	Turbine final output power		v_str
wref	$\omega_{ref}$	Algeb	Speed reference variable		v_str
pref	$P_{ref}$	Algeb	Reference power input		v_str
wd	$\omega_{dev}$	Algeb	Generator speed deviation	p.u.	v_str
pd	$P_d$	Algeb	Pref plus speed deviation times gain	p.u.	v_str
LL_y	$y_{LL}$	Algeb	Output of lead-lag		v_str
DB_y	$y_{DB}$	Algeb	Deadband type 1 output		v_str
tm	$\tau_m$	ExtAlgeb	Mechanical power interface to SynGen		

## Initialization Equations

Name	Symbol	Type	Initial Value
LAG_y	$y_{LAG}$	State	$P_d$
LL_x	$x'_{LL}$	State	$y_{LAG}$
omega	$\omega$	ExtState	
paux	$P_{aux}$	Algeb	$P_{aux0}$
pout	$P_{out}$	Algeb	$\tau_{m0}u_e$
wref	$\omega_{ref}$	Algeb	$\omega_{ref0}$
pref	$P_{ref}$	Algeb	$\tau_{m0}$
wd	$\omega_{dev}$	Algeb	0
pd	$P_d$	Algeb	$\tau_{m0}u_e$
LL_y	$y_{LL}$	Algeb	$y_{LAG}$
DB_y	$y_{DB}$	Algeb	$1.0z_l^{db_{DB}}(\omega_{dev} - db_L) + 1.0z_u^{db_{DB}}(\omega_{dev} - db_U)$
tm	$\tau_m$	ExtAlgeb	

## Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
LAG_y	$y_{LAG}$	State	$P_d - y_{LAG}$	$T_1$
LL_x	$x'_{LL}$	State	$-x'_{LL} + y_{LAG}$	$T_3$
omega	$\omega$	ExtState	0	

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
paux	$P_{aux}$	Algeb	$P_{aux0} - P_{aux}$
pout	$P_{out}$	Algeb	$-D_t y_{DB} - P_{out} + y_{LL}$
wref	$\omega_{ref}$	Algeb	$\omega_{ref0} - \omega_{ref}$
pref	$P_{ref}$	Algeb	$P_{ref0} - P_{ref}$
wd	$\omega_{dev}$	Algeb	$-\omega_{dev} + u_e (\omega - \omega_{ref})$
pd	$P_d$	Algeb	$-P_d + u_e (Gy_{DB} + P_{aux} + P_{ref})$
LL_y	$y_{LL}$	Algeb	$T_2(-x'_{LL} + y_{LAG}) + T_3x'_{LL} - T_3y_{LL} + (z_1^{LT_{LL}})^2(-x'_{LL} + y_{LL})$
DB_y	$y_{DB}$	Algeb	$-y_{DB} + 1.0z_l^{db_{DB}}(\omega_{dev} - db_L) + 1.0z_u^{db_{DB}}(\omega_{dev} - db_U)$
tm	$\tau_m$	ExtAlgeb	$u_e (P_{out} - \tau_{m0})$

## Services

Name	Symbol	Equation	Type
ue	$u_e$	$uu_g$	ConstService
pref0	$P_{ref0}$	$\tau_{m0}$	ConstService
paux0	$P_{aux0}$	0	ConstService
gain	$G$	$\frac{u_e}{R}$	ConstService

## Discretes

Name	Symbol	Type	Info
LAG_lim	$lim_{LAG}$	AntiWindup	Limiter in Lag
LL_LT1	$LT_{LL}$	LessThan	
LL_LT2	$LT_{LL}$	LessThan	
DB_db	$db_{DB}$	DeadBand	

## Blocks

Name	Symbol	Type	Info
LAG	$LAG$	LagAntiWindup	
LL	$LL$	LeadLag	
DB	$DB$	DeadBand1	deadband for speed deviation

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)

### 5.35.6 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,  $K1 + K2 + \dots + K8 = 1.0$ . 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	<i>u</i>	connection status	1	bool	
name		device name			
syn		Synchronous generator idx			mandatory,unique
Tn	<i>T<sub>n</sub></i>	Turbine power rating. Equal to <i>Sn</i> if not provided.	0	MVA	
wref0	$\omega_{ref0}$	Base speed reference	1	p.u.	
syn2		Optional SynGen idx			
K	<i>K</i>	Gain (1/R) in mach. base	20	p.u. (power)	power
T1	<i>T<sub>1</sub></i>	Gov. lag time const.	1		
T2	<i>T<sub>2</sub></i>	Gov. lead time const.	1		
T3	<i>T<sub>3</sub></i>	Valve controller time const.	0.100		
UO	<i>U<sub>o</sub></i>	Max. valve opening rate	0.100	p.u./sec	
UC	<i>U<sub>c</sub></i>	Max. valve closing rate	-0.100	p.u./sec	
PMAX	<i>P<sub>MAX</sub></i>	Max. turbine power	5		power
PMIN	<i>P<sub>MIN</sub></i>	Min. turbine power	0		power
T4	<i>T<sub>4</sub></i>	Inlet piping/steam bowl time constant	0.400		
K1	<i>K<sub>1</sub></i>	Fraction of power from HP	0.500		
K2	<i>K<sub>2</sub></i>	Fraction of power from LP	0		
T5	<i>T<sub>5</sub></i>	Time constant of 2nd boiler pass	8		
K3	<i>K<sub>3</sub></i>	Fraction of HP shaft power after 2nd boiler pass	0.500		
K4	<i>K<sub>4</sub></i>	Fraction of LP shaft power after 2nd boiler pass	0		
T6	<i>T<sub>6</sub></i>	Time constant of 3rd boiler pass	0.500		
K5	<i>K<sub>5</sub></i>	Fraction of HP shaft power after 3rd boiler pass	0		

continues on next page

Table 46 – continued from previous page

Name	Symbol	Description	Default	Unit	Properties
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		
Sg	$S_n$	Rated power from generator	0	MVA	
ug	$u_g$	Generator connection status	0	bool	
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	$x'_{LL}$	State	State in lead-lag	v_str	
IAW_y	$y_{IAW}$	State	AW Integrator output	v_str	
L4_y	$y_{L4}$	State	State in lag transfer function	v_str	
L5_y	$y_{L5}$	State	State in lag transfer function	v_str	
L6_y	$y_{L6}$	State	State in lag transfer function	v_str	
L7_y	$y_{L7}$	State	State in lag transfer function	v_str	
omega	$\omega$	ExtState	Generator speed	p.u.	
paux	$P_{aux}$	Algeb	Auxiliary power input	v_str	
pout	$P_{out}$	Algeb	Turbine final output power	v_str	
wref	$\omega_{ref}$	Algeb	Speed reference variable	v_str	
wd	$\omega_{dev}$	Algeb	Generator under speed	p.u.	v_str
LL_y	$y_{LL}$	Algeb	Output of lead-lag	v_str	
vs	$V_s$	Algeb	Valve speed	v_str	
vsl	$V_{sl}$	Algeb	Valve move speed after limiter	v_str	
PHP	$P_{HP}$	Algeb	HP output	v_str	
PLP	$P_{LP}$	Algeb	LP output	v_str	
tm	$\tau_m$	ExtAlgeb	Mechanical power interface to SynGen		
tm2	$\tau_{m2}$	ExtAlgeb	Mechanical power to syn2		

## Initialization Equations

Name	Symbol	Type	Initial Value
LL_x	$x'_{LL}$	State	$\omega_{dev}$
IAW_y	$y_{IAW}$	State	$t m_{012}$
L4_y	$y_{L4}$	State	$y_{IAW}$
L5_y	$y_{L5}$	State	$y_{L4}$
L6_y	$y_{L6}$	State	$y_{L5}$
L7_y	$y_{L7}$	State	$y_{L6}$
omega	$\omega$	ExtState	
paux	$P_{aux}$	Algeb	$P_{aux0}$
pout	$P_{out}$	Algeb	$\tau_{m0} u_e$
wref	$\omega_{ref}$	Algeb	$\omega_{ref0}$
wd	$\omega_{dev}$	Algeb	0
LL_y	$y_{LL}$	Algeb	$\omega_{dev}$
vs	$V_s$	Algeb	0
vsl	$V_{sl}$	Algeb	$U_c z_l^{HL} + U_o z_u^{HL} + V_s z_i^{HL}$
PHP	$P_{HP}$	Algeb	$u_e (K_{1n} y_{L4} + K_{3n} y_{L5} + K_{5n} y_{L6} + K_{7n} y_{L7})$
PLP	$P_{LP}$	Algeb	$u_e (K_{2n} y_{L4} + K_{4n} y_{L5} + K_{6n} y_{L6} + K_{8n} y_{L7})$
tm	$\tau_m$	ExtAlgeb	
tm2	$\tau_{m2}$	ExtAlgeb	

## Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
LL_x	$x'_{LL}$	State	$\omega_{dev} - x'_{LL}$	$T_1$
IAW_y	$y_{IAW}$	State	$V_{sl}$	1
L4_y	$y_{L4}$	State	$y_{IAW} - y_{L4}$	$T_4$
L5_y	$y_{L5}$	State	$y_{L4} - y_{L5}$	$T_5$
L6_y	$y_{L6}$	State	$y_{L5} - y_{L6}$	$T_6$
L7_y	$y_{L7}$	State	$y_{L6} - y_{L7}$	$T_7$
omega	$\omega$	ExtState	0	

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
paux	$P_{aux}$	Algeb	$P_{aux0} - P_{aux}$
pout	$P_{out}$	Algeb	$P_{HP}u_e - P_{out}$
wref	$\omega_{ref}$	Algeb	$\omega_{ref0} - \omega_{ref}$
wd	$\omega_{dev}$	Algeb	$-\omega_{dev} + u_e (-\omega + \omega_{ref})$
LL_y	$y_{LL}$	Algeb	$KT_1x'_{LL} + KT_2(\omega_{dev} - x'_{LL}) - T_1y_{LL} + (z_1^{LT_{LL}})^2(-Kx'_{LL} + y_{LL})$
vs	$V_s$	Algeb	$-V_s + \frac{u_e(P_{aux} + tm012 - y_{IAW} + y_{LL})}{T_3}$
vsl	$V_{sl}$	Algeb	$U_c z_l^{HL} + U_o z_u^{HL} + V_s z_i^{HL} - V_{sl}$
PHP	$P_{HP}$	Algeb	$-P_{HP} + u_e(K_{1n}y_{L4} + K_{3n}y_{L5} + K_{5n}y_{L6} + K_{7n}y_{L7})$
PLP	$P_{LP}$	Algeb	$-P_{LP} + u_e(K_{2n}y_{L4} + K_{4n}y_{L5} + K_{6n}y_{L6} + K_{8n}y_{L7})$
tm	$\tau_m$	ExtAlgeb	$u_e(P_{out} - \tau_{m0})$
tm2	$\tau_{m2}$	ExtAlgeb	$u_e z_{syn2}(P_{LP} - \tau_{m02})$

## Services

Name	Symbol	Equation	Type
ue	$u_e$	$uu_g$	ConstService
pref0	$P_{ref0}$	$\tau_{m0}$	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	$K_{coeff}$	$\frac{1}{\sum_{i=1}^8 K_i}$	ConstService
K1n	$K_{1n}$	$K_1 K_{coeff}$	ConstService
K2n	$K_{2n}$	$K_2 K_{coeff}$	ConstService
K3n	$K_{3n}$	$K_3 K_{coeff}$	ConstService
K4n	$K_{4n}$	$K_4 K_{coeff}$	ConstService
K5n	$K_{5n}$	$K_5 K_{coeff}$	ConstService
K6n	$K_{6n}$	$K_6 K_{coeff}$	ConstService
K7n	$K_{7n}$	$K_7 K_{coeff}$	ConstService
K8n	$K_{8n}$	$K_8 K_{coeff}$	ConstService
_tm0K2	$tm0K2$	$\tau_{m0} z_{syn2}(K_{2n} + K_{4n} + K_{6n} + K_{8n})$	PostInitService
_tm02K1	$tm02K1$	$\tau_{m02}(K_{1n} + K_{3n} + K_{5n} + K_{7n})$	PostInitService
tm012	$tm012$	$\tau_{m02} + \tau_{m0}$	ConstService

## Discretes

Name	Symbol	Type	Info
LL_LT1	$LT_{LL}$	LessThan	
LL_LT2	$LT_{LL}$	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

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)

## 5.35.7 IEESGO

IEEE Standard Governor (IEESGO).

## Parameters

Name	Symbol	Description	De-fault	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 $S_n$ if not provided.	0	MVA	
wref0	$\omega_{ref0}$	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	
Sg	$S_n$	Rated power from generator	0	MVA	
ug	$u_g$	Generator connection status	0	bool	
Vn	$V_n$	Rated voltage from generator	0	kV	

## Variables

Name	Symbol	Type	Description	Unit	Properties
F1_y	$y_{F1}$	State	State in lag transfer function	v_str	
F2_x	$x'_{F2}$	State	State in lead-lag	v_str	
F3_y	$y_{F3}$	State	State in lag transfer function	v_str	
F4_y	$y_{F4}$	State	State in lag transfer function	v_str	
F5_y	$y_{F5}$	State	State in lag transfer function	v_str	
omega	$\omega$	ExtState	Generator speed	p.u.	
paux	$P_{aux}$	Algeb	Auxiliary power input	v_str	
pout	$P_{out}$	Algeb	Turbine final output power	v_str	
wref	$\omega_{ref}$	Algeb	Speed reference variable	v_str	
F2_y	$y_{F2}$	Algeb	Output of lead-lag	v_str	
HL_x	$x_{HL}$	Algeb	Value before limiter	v_str	
HL_y	$y_{HL}$	Algeb	Output after limiter and post gain	v_str	
tm	$\tau_m$	ExtAlgeb	Mechanical power interface to SynGen		

## Initialization Equations

Name	Symbol	Type	Initial Value
F1_y	$y_{F1}$	State	$K_1 u_e (\omega - \omega_{ref})$
F2_x	$x'_{F2}$	State	$y_{F1}$
F3_y	$y_{F3}$	State	$1.0 y_{HL}$
F4_y	$y_{F4}$	State	$K_2 y_{F3}$
F5_y	$y_{F5}$	State	$K_3 y_{F4}$
omega	$\omega$	ExtState	
paux	$P_{aux}$	Algeb	$P_{aux0}$
pout	$P_{out}$	Algeb	$\tau_{m0} u_e$
wref	$\omega_{ref}$	Algeb	$\omega_{ref0}$
F2_y	$y_{F2}$	Algeb	$y_{F1}$
HL_x	$x_{HL}$	Algeb	$1.0 u_e (P_{aux} + P_{ref0} - y_{F2})$
HL_y	$y_{HL}$	Algeb	$1.0 P_{MAX} z_u^{lim_{HL}} + 1.0 P_{MIN} z_l^{lim_{HL}} + 1.0 x_{HL} z_i^{lim_{HL}}$
tm	$\tau_m$	ExtAlgeb	

## Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
F1_y	$y_{F1}$	State	$K_1 u_e (\omega - \omega_{ref}) - y_{F1}$	$T_1$
F2_x	$x'_{F2}$	State	$-x'_{F2} + y_{F1}$	$T_3$
F3_y	$y_{F3}$	State	$-y_{F3} + 1.0 y_{HL}$	$T_4$
F4_y	$y_{F4}$	State	$K_2 y_{F3} - y_{F4}$	$T_5$
F5_y	$y_{F5}$	State	$K_3 y_{F4} - y_{F5}$	$T_6$
omega	$\omega$	ExtState	0	

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
paux	$P_{aux}$	Algeb	$P_{aux0} - P_{aux}$
pout	$P_{out}$	Algeb	$-P_{out} + u_e (y_{F3} \cdot (1 - K_2) + y_{F4} \cdot (1 - K_3) + y_{F5})$
wref	$\omega_{ref}$	Algeb	$\omega_{ref0} - \omega_{ref}$
F2_y	$y_{F2}$	Algeb	$1.0 T_2 (-x'_{F2} + y_{F1}) + 1.0 T_3 x'_{F2} - T_3 y_{F2} + (z_1^{LT_{F2}})^2 (-1.0 x'_{F2} + y_{F2})$
HL_x	$x_{HL}$	Algeb	$1.0 u_e (P_{aux} + P_{ref0} - y_{F2}) - x_{HL}$
HL_y	$y_{HL}$	Algeb	$1.0 P_{MAX} z_u^{lim_{HL}} + 1.0 P_{MIN} z_l^{lim_{HL}} + 1.0 x_{HL} z_i^{lim_{HL}} - y_{HL}$
tm	$\tau_m$	ExtAlgeb	$u_e (P_{out} - \tau_{m0})$

## Services

Name	Symbol	Equation	Type
ue	$u_e$	$uu_g$	ConstService
pref0	$P_{ref0}$	$\tau_{m0}$	ConstService
paux0	$P_{aux0}$	0	ConstService

## Discretes

Name	Symbol	Type	Info
F2_LT1	$LT_{F2}$	LessThan	
F2_LT2	$LT_{F2}$	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	

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)

### 5.35.8 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](https://www.neplan.ch/wp-content/uploads/2015/08/Nep_TURBINES_GOV.pdf)

#### Parameters

Name	Symbol	Description	De-fault	Unit	Properties
idx		unique device idx			
u	$u$	connection status	1	<i>bool</i>	
name		device name			
syn		Synchronous generator idx			mandatory,unique
Tn	$T_n$	Turbine power rating. Equal to $S_n$ if not provided.	0	MVA	
wref0	$\omega_{ref0}$	Base speed reference	1	<i>p.u.</i>	
R	$R$	Speed regulation gain (mach. base default)	0.050	<i>p.u.</i>	ipower
VMAX	$V_{max}$	Maximum valve position	1.200	<i>p.u.</i>	power
VMIN	$V_{min}$	Minimum valve position	0	<i>p.u.</i>	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
Sg	$S_n$	Rated power from generator	0	MVA	
ug	$u_g$	Generator connection status	0	<i>bool</i>	
Vn	$V_n$	Rated voltage from generator	0	kV	

## Variables

Name	Symbol	Type	Description	Unit	Properties
LAG_y	$y_{LAG}$	State	State in lag TF		v_str
LG2_y	$y_{LG2}$	State	State in lag transfer function		v_str
LG3_y	$y_{LG3}$	State	State in lag transfer function		v_str
omega	$\omega$	ExtState	Generator speed	p.u.	
paux	$P_{aux}$	Algeb	Auxiliary power input		v_str
pout	$P_{out}$	Algeb	Turbine final output power		v_str
wref	$\omega_{ref}$	Algeb	Speed reference variable		v_str
pref	$P_{ref}$	Algeb	Reference power input		v_str
wd	$\omega_{dev}$	Algeb	Generator under speed	p.u.	v_str
pd	$P_d$	Algeb	Pref plus under speed times gain	p.u.	v_str
v9	$V_9$	Algeb	$V_9$ for LVGate input		v_str
LVG_y	$y_{LVG}$	Algeb	LVGate output		v_str
tm	$\tau_m$	ExtAlgeb	Mechanical power interface to SynGen		

## Initialization Equations

Name	Symbol	Type	Initial Value
LAG_y	$y_{LAG}$	State	$y_{LVG}$
LG2_y	$y_{LG2}$	State	$y_{LAG}$
LG3_y	$y_{LG3}$	State	$y_{LG2}$
omega	$\omega$	ExtState	
paux	$P_{aux}$	Algeb	$P_{aux0}$
pout	$P_{out}$	Algeb	$\tau_{m0}u_e$
wref	$\omega_{ref}$	Algeb	$\omega_{ref0}$
pref	$P_{ref}$	Algeb	$R\tau_{m0}$
wd	$\omega_{dev}$	Algeb	0
pd	$P_d$	Algeb	$\tau_{m0}u_e$
v9	$V_9$	Algeb	$u_e(A_T + K_T(A_T - \tau_{m0}))$
LVG_y	$y_{LVG}$	Algeb	$P_d z_1^{None_{LVG}} + V_9 z_0^{None_{LVG}}$
tm	$\tau_m$	ExtAlgeb	

## Differential Equations

Name	Symbol	Type	RHS of Equation "T x' = f(x, y)"	T (LHS)
LAG_y	$y_{LAG}$	State	$-y_{LAG} + y_{LVG}$	$T_1$
LG2_y	$y_{LG2}$	State	$y_{LAG} - y_{LG2}$	$T_2$
LG3_y	$y_{LG3}$	State	$y_{LG2} - y_{LG3}$	$T_3$
omega	$\omega$	ExtState	0	

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
paux	$P_{aux}$	Algeb	$P_{aux0} - P_{aux}$
pout	$P_{out}$	Algeb	$-P_{out} + u_e (-D_t \omega_{dev} + y_{LG2})$
wref	$\omega_{ref}$	Algeb	$\omega_{ref0} - \omega_{ref}$
pref	$P_{ref}$	Algeb	$P_{ref0}R - P_{ref}$
wd	$\omega_{dev}$	Algeb	$-\omega_{dev} + u_e (\omega - \omega_{ref})$
pd	$P_d$	Algeb	$G u_e (P_{aux} + P_{ref} - \omega_{dev}) - P_d$
v9	$V_9$	Algeb	$-V_9 + u_e (A_T + K_T (A_T - y_{LG3}))$
LVG_y	$y_{LVG}$	Algeb	$P_d z_1^{None_{LVG}} + V_9 z_0^{None_{LVG}} - y_{LVG}$
tm	$\tau_m$	ExtAlgeb	$u_e (P_{out} - \tau_{m0})$

## Services

Name	Symbol	Equation	Type
ue	$u_e$	$uu_g$	ConstService
pref0	$P_{ref0}$	$\tau_{m0}$	ConstService
paux0	$P_{aux0}$	0	ConstService
gain	$G$	$\frac{u_e}{R}$	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	<i>LVG</i>	LVGate	LVGate
LAG	<i>LAG</i>	LagAntiWindup	
LG2	<i>LG2</i>	Lag	Lag T2
LG3	<i>LG3</i>	Lag	Lag T3

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)

### 5.35.9 HYG OV

HYGOV turbine governor model.

Implements the PSS/E HYG OV model without deadband.

Reference:

[1] PSSE, Model Library, HYG OV

## 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 $S_n$ if not provided.	0	MVA	
wref0	$\omega_{ref0}$	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		
Sg	$S_n$	Rated power from generator	0	MVA	
ug	$u_g$	Generator connection status	0	bool	
Vn	$V_n$	Rated voltage from generator	0	kV	

## Variables

Name	Symbol	Type	Description	Unit	Properties
LG_y	$y_{LG}$	State	State in lag transfer function		v_str
gtpos	$\delta$	State	State in gate position (c)	rad	v_str
LAG_y	$y_{LAG}$	State	State in lag transfer function		v_str
q_y	$y_q$	State	Integrator output		v_str
omega	$\omega$	ExtState	Generator speed	p.u.	
paux	$P_{aux}$	Algeb	Auxiliary power input		v_str
pout	$P_{out}$	Algeb	Turbine final output power		v_str
wref	$\omega_{ref}$	Algeb	Speed reference variable		v_str
pref	$P_{ref}$	Algeb	Reference power input		v_str
wd	$\omega_{dev}$	Algeb	Generator speed deviation	p.u.	v_str
pd	$P_d$	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	$\tau_m$	ExtAlgeb	Mechanical power interface to SynGen		

## Initialization Equations

Name	Symbol	Type	Initial Value
LG_y	$y_{LG}$	State	$P_d$
gtpos	$\delta$	State	$q_0$
LAG_y	$y_{LAG}$	State	$dg$
q_y	$y_q$	State	$q_0$
omega	$\omega$	ExtState	
paux	$P_{aux}$	Algeb	$P_{aux0}$
pout	$P_{out}$	Algeb	$\tau_m u_e$
wref	$\omega_{ref}$	Algeb	$\omega_{ref0}$
pref	$P_{ref}$	Algeb	$Rq_0$
wd	$\omega_{dev}$	Algeb	0
pd	$P_d$	Algeb	0
dg	$dg$	Algeb	$q_0$
h	$h$	Algeb	1
tm	$\tau_m$	ExtAlgeb	

## Differential Equations

Name	Symbol	Type	RHS of Equation "T' x' = f(x, y)"	T (LHS)
LG_y	$y_{LG}$	State	$P_d - y_{LG}$	$T_f$
gtpos	$\delta$	State	$y_{LG}$	
LAG_y	$y_{LAG}$	State	$dg - y_{LAG}$	$T_g$
q_y	$y_q$	State	$1 - \frac{y_q^2}{y_{LAG}^2}$	$T_w$
omega	$\omega$	ExtState	0	

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
paux	$P_{aux}$	Algeb	$P_{aux0} - P_{aux}$
pout	$P_{out}$	Algeb	$-P_{out} + u_e (A_t h (-q_{NL} + y_q) - D_t \omega_{dev} y_{LAG})$
wref	$\omega_{ref}$	Algeb	$\omega_{ref0} - \omega_{ref}$
pref	$P_{ref}$	Algeb	$-P_{ref} + Rq_0$
wd	$\omega_{dev}$	Algeb	$-\omega_{dev} + u_e (\omega - \omega_{ref})$
pd	$P_d$	Algeb	$-P_d + u_e (P_{aux} + P_{ref} - Rdg - \omega_{dev})$
dg	$dg$	Algeb	$1/ry_{LG} + \delta - dg$
h	$h$	Algeb	$-h + \frac{y_q^2}{y_{LAG}^2}$
tm	$\tau_m$	ExtAlgeb	$u_e (P_{out} - \tau_{m0})$

## Services

Name	Symbol	Equation	Type
ue	$u_e$	$uu_g$	ConstService
pref0	$P_{ref0}$	$\tau_{m0}$	ConstService
paux0	$P_{aux0}$	0	ConstService
VELMn	$-VELM$	$-VELM$	ConstService
tr	$r * Tr$	$T_r r$	ConstService
gr	$1/r$	$\frac{1}{r}$	ConstService
ratel	$rate_l$	$-1/r - VELM$	ConstService
rateu	$rate_u$	$-1/r + VELM$	ConstService
q0	$q_0$	$q_{NL} + \frac{\tau_{m0}}{A_t}$	ConstService
dgl	$dg_{lower}$	$-1/ry_{LG} - VELM$	VarService
dgu	$dg_{upper}$	$-1/ry_{LG} + VELM$	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)

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)

### 5.35.10 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 $S_n$ if not provided.	0	MVA	
wref0	$\omega_{ref0}$	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.	
Sg	$S_n$	Rated power from generator	0	MVA	
ug	$u_g$	Generator connection status	0	bool	
Vn	$V_n$	Rated voltage from generator	0	kV	

## Variables

Name	Symbol	Type	Description	Unit	Properties
LG_y	$y_{LG}$	State	State in lag transfer function		v_str
gtpos	$\delta$	State	State in gate position (c)	rad	v_str
LAG_y	$y_{LAG}$	State	State in lag transfer function		v_str
q_y	$y_q$	State	Integrator output		v_str
omega	$\omega$	ExtState	Generator speed	p.u.	
paux	$P_{aux}$	Algeb	Auxiliary power input		v_str
pout	$P_{out}$	Algeb	Turbine final output power		v_str
wref	$\omega_{ref}$	Algeb	Speed reference variable		v_str
pref	$P_{ref}$	Algeb	Reference power input		v_str
wd	$\omega_{dev}$	Algeb	Generator speed deviation	p.u.	v_str
pd	$P_d$	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	$y_{DB}$	Algeb	Deadband type 1 output		v_str
tm	$\tau_m$	ExtAlgeb	Mechanical power interface to SynGen		

## Initialization Equations

Name	Symbol	Type	Initial Value
LG_y	$y_{LG}$	State	$P_d$
gtpos	$\delta$	State	$q_0$
LAG_y	$y_{LAG}$	State	$dg$
q_y	$y_q$	State	$q_0$
omega	$\omega$	ExtState	
paux	$P_{aux}$	Algeb	$P_{aux0}$
pout	$P_{out}$	Algeb	$\tau_{m0} u_e$
wref	$\omega_{ref}$	Algeb	$\omega_{ref0}$
pref	$P_{ref}$	Algeb	$Rq_0$
wd	$\omega_{dev}$	Algeb	0
pd	$P_d$	Algeb	0
dg	$dg$	Algeb	$q_0$
h	$h$	Algeb	1
DB_y	$y_{DB}$	Algeb	$1.0 z_l^{db_{DB}} (\omega_{dev} - db_L) + 1.0 z_u^{db_{DB}} (\omega_{dev} - db_U)$
tm	$\tau_m$	ExtAlgeb	

## Differential Equations

Name	Symbol	Type	RHS of Equation "T' x' = f(x, y)"	T (LHS)
LG_y	$y_{LG}$	State	$P_d - y_{LG}$	$T_f$
gtpos	$\delta$	State	$y_{LG}$	
LAG_y	$y_{LAG}$	State	$dg - y_{LAG}$	$T_g$
q_y	$y_q$	State	$1 - \frac{y_q^2}{y_{LAG}^2}$	$T_w$
omega	$\omega$	ExtState	0	

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
paux	$P_{aux}$	Algeb	$P_{aux0} - P_{aux}$
pout	$P_{out}$	Algeb	$-P_{out} + u_e (A_t h (-q_{NL} + y_q) - D_t \omega_{dev} y_{LAG})$
wref	$\omega_{ref}$	Algeb	$\omega_{ref0} - \omega_{ref}$
pref	$P_{ref}$	Algeb	$-P_{ref} + R q_0$
wd	$\omega_{dev}$	Algeb	$-\omega_{dev} + u_e (\omega - \omega_{ref})$
pd	$P_d$	Algeb	$-P_d + u_e (P_{aux} + P_{ref} - R d g - y_{DB})$
dg	$d g$	Algeb	$1/r y_{LG} + \delta - d g$
h	$h$	Algeb	$-h + \frac{y_q^2}{y_{LAG}^2}$
DB_y	$y_{DB}$	Algeb	$-y_{DB} + 1.0 z_l^{db_{DB}} (\omega_{dev} - db_L) + 1.0 z_u^{db_{DB}} (\omega_{dev} - db_U)$
tm	$\tau_m$	ExtAlgeb	$u_e (P_{out} - \tau_{m0})$

## Services

Name	Symbol	Equation	Type
ue	$u_e$	$u u_g$	ConstService
pref0	$P_{ref0}$	$\tau_{m0}$	ConstService
paux0	$P_{aux0}$	0	ConstService
VELMn	$-VELM$	$-VELM$	ConstService
tr	$r * Tr$	$T_r r$	ConstService
gr	$1/r$	$\frac{1}{r}$	ConstService
ratel	$rate_l$	$-1/r - VELM$	ConstService
rateu	$rate_u$	$-1/r + VELM$	ConstService
q0	$q_0$	$q_{NL} + \frac{\tau_{m0}}{A_t}$	ConstService
dgl	$d g_{lower}$	$-1/r y_{LG} - VELM$	VarService
dgu	$d g_{upper}$	$-1/r y_{LG} + VELM$	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

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)

### 5.35.11 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	De-fault	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 $S_n$ if not provided.	0	MVA	
wref0	$\omega_{ref0}$	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 ( $R_{temp} < R_{perm}$ )	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
Sg	$S_n$	Rated power from generator	0	MVA	
ug	$u_g$	Generator connection status	0	bool	
Vn	$V_n$	Rated voltage from generator	0	kV	

## Variables

Name	Symbol	Type	Description	Unit	Properties
GATE_y	$y_{GATE}$	State	AW Integrator output		v_str
WO_x	$x'_{WO}$	State	State in washout filter		v_str
LAG_y	$y_{LAG}$	State	State in lag transfer function		v_str
q_y	$y_q$	State	Integrator output		v_str
omega	$\omega$	ExtState	Generator speed	p.u.	
paux	$P_{aux}$	Algeb	Auxiliary power input		v_str
pout	$P_{out}$	Algeb	Turbine final output power		v_str
wref	$\omega_{ref}$	Algeb	Speed reference variable		v_str
pref	$P_{ref}$	Algeb	Reference power input		v_str
wd	$\omega_{dev}$	Algeb	Generator speed deviation	p.u.	v_str
SV_x	$x_{SV}$	Algeb	Value before limiter		v_str
SV_y	$y_{SV}$	Algeb	Output after limiter and post gain		v_str
WO_y	$y_{WO}$	Algeb	Output of washout filter		v_str
Psum	$P_{sum}$	Algeb	summation of power input to servo	p.u.	v_str
trhead	$trhead$	Algeb	turbine head	p.u.	v_str
tm	$\tau_m$	ExtAlgeb	Mechanical power interface to SynGen		

## Initialization Equations

Name	Symbol	Type	Initial Value
GATE_y	$y_{GATE}$	State	$\frac{q_0}{H_{dam}^{0.5}}$
WO_x	$x'_{WO}$	State	$y_{GATE}$
LAG_y	$y_{LAG}$	State	$P_{sum}$
q_y	$y_q$	State	$q_0$
omega	$\omega$	ExtState	
paux	$P_{aux}$	Algeb	$P_{aux0}$
pout	$P_{out}$	Algeb	$\tau_m u_e$
wref	$\omega_{ref}$	Algeb	$\omega_{ref0}$
pref	$P_{ref}$	Algeb	$\frac{R_{perm} q_0}{H_{dam}^{0.5}}$
wd	$\omega_{dev}$	Algeb	0
SV_x	$x_{SV}$	Algeb	$1/T_g y_{LAG}$
SV_y	$y_{SV}$	Algeb	$U_C z_l^{lim_{SV}} + U_O z_u^{lim_{SV}} + x_{SV} z_i^{lim_{SV}}$
WO_y	$y_{WO}$	Algeb	0
Psum	$P_{sum}$	Algeb	0
trhead	$trhead$	Algeb	$H_{dam}$
tm	$\tau_m$	ExtAlgeb	

## Differential Equations

Name	Symbol	Type	RHS of Equation "T' x' = f(x, y)"	T (LHS)
GATE_y	$y_{GATE}$	State	$y_{SV}$	1
WO_x	$x'_{WO}$	State	$-x'_{WO} + y_{GATE}$	$T_r$
LAG_y	$y_{LAG}$	State	$P_{sum} - y_{LAG}$	$T_p$
q_y	$y_q$	State	$H_{dam} - trhead$	$T_w$
omega	$\omega$	ExtState	0	

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
paux	$P_{aux}$	Algeb	$P_{aux0} - P_{aux}$
pout	$P_{out}$	Algeb	$-P_{out} + u_e (A_t trhead (-q_{NL} + y_q) - D_{turb} \omega_{dev} y_{GATE})$
wref	$\omega_{ref}$	Algeb	$\omega_{ref0} - \omega_{ref}$
pref	$P_{ref}$	Algeb	$\frac{R_{perm} q_0}{H_{dam}^{0.5}} - P_{ref}$
wd	$\omega_{dev}$	Algeb	$-\omega_{dev} + u_e (\omega - \omega_{ref})$
SV_x	$x_{SV}$	Algeb	$1/T_g y_{LAG} - x_{SV}$
SV_y	$y_{SV}$	Algeb	$U_C z_l^{lim_{SV}} + U_O z_u^{lim_{SV}} + x_{SV} z_i^{lim_{SV}} - y_{SV}$
WO_y	$y_{WO}$	Algeb	$R_{temp} * T_r (-x'_{WO} + y_{GATE}) - T_r y_{WO}$
Psum	$P_{sum}$	Algeb	$-P_{sum} + u_e (P_{aux} + P_{ref} - R_{perm} y_{GATE} - \omega_{dev} - y_{WO})$
trhead	$trhead$	Algeb	$-trhead + \frac{y_q^2}{y_{GATE}^2}$
tm	$\tau_m$	ExtAlgeb	$u_e (P_{out} - \tau_{m0})$

## Services

Name	Symbol	Equation	Type
ue	$u_e$	$uu_g$	ConstService
pref0	$P_{ref0}$	$\tau_{m0}$	ConstService
paux0	$P_{aux0}$	0	ConstService
iTg	$1/T_g$	$\frac{u}{T_g}$	ConstService
R	$R_{temp} + R_{perm}$	$R_{perm} + R_{temp}$	ConstService
TrRtemp	$R_{temp} * T_r$	$R_{temp} T_r$	ConstService
q0	$q_0$	$q_{NL} + \frac{\tau_{m0}}{A_t H_{dam}}$	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 ( $q$ )

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)

## 5.36 Undefined

The undefined group. Holds models with no group.

Common Parameters: u, name

## 5.37 VoltComp

Voltage compensator group for synchronous generators.

Common Parameters: u, name, rc, xc

Common Variables: vcomp

Available models: [IEEEVVC](#)

### 5.37.1 IEEEVVC

Voltage compensator IEEEVVC model.

Reference:

[1] PowerWorld, Voltage Compensator, IEEEVVC, [Online],

[2] NEPLAN, Exciters Models, [Online],

Available:

[https://www.powerworld.com/WebHelp/Content/TransientModels\\_HTML/Voltage%20Compensator%20IEEEVVC.htm?TocPath=%7C%7C%7CIEEEVVC%7C\\_\\_\\_\\_\\_0](https://www.powerworld.com/WebHelp/Content/TransientModels_HTML/Voltage%20Compensator%20IEEEVVC.htm?TocPath=%7C%7C%7CIEEEVVC%7C_____0)

[https://www.neplan.ch/wp-content/uploads/2015/08/Nep\\_EXCITERS1.pdf](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	
syn		Retrieved generator idx	0		

#### Variables

Name	Symbol	Type	Description	Unit	Properties
vcomp	$v_{comp}$	Algeb	Compensator output voltage to exciter		v_str
v	$V$	ExtAlgeb	Retrieved bus terminal voltage		
vd	$V_d$	ExtAlgeb	d-axis machine voltage		
vq	$V_q$	ExtAlgeb	q-axis machine voltage		
Id	$I_d$	ExtAlgeb	d-axis machine current		
Iq	$I_q$	ExtAlgeb	q-axis machine current		
Eterm	$Eterm$	ExtAlgeb			v_str

## Initialization Equations

Name	Symbol	Type	Initial Value
vcomp	$v_{comp}$	Algeb	$-Vu + V_{CT}$
v	$V$	ExtAlgeb	
vd	$V_d$	ExtAlgeb	
vq	$V_q$	ExtAlgeb	
Id	$I_d$	ExtAlgeb	
Iq	$I_q$	ExtAlgeb	
Eterm	$Eterm$	ExtAlgeb	$v_{comp}$

## Algebraic Equations

Name	Symbol	Type	RHS of Equation "0 = g(x, y)"
vcomp	$v_{comp}$	Algeb	$-Vu + V_{CT} - v_{comp}$
v	$V$	ExtAlgeb	0
vd	$V_d$	ExtAlgeb	0
vq	$V_q$	ExtAlgeb	0
Id	$I_d$	ExtAlgeb	0
Iq	$I_q$	ExtAlgeb	0
Eterm	$Eterm$	ExtAlgeb	$v_{comp}$

## Services

Name	Symbol	Equation	Type
vct	$V_{CT}$	$u  V_d + iV_q + (I_d + iI_q) (r_c + ix_c) $	VarService

Config Fields in [IEEEVVC]

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)



## API REFERENCE

### 6.1 System

<code>andes.system</code>	System class for power system data and methods
<code>andes.variables</code>	Variable package with classes for numerical DAE.

#### 6.1.1 andes.system

System class for power system data and methods

##### Functions

<code>example([setup, no_output])</code>	Return an <code>andes.system.System</code> object for the <code>ieee14_linetrip.xlsx</code> as an example.
<code>fix_view_arrays(system)</code>	Point NumPy arrays without OWNDATA (termed "view arrays" here) to the source array.
<code>import_pycode([user_pycode_path])</code>	Helper function to import generated pycode in the following priority:
<code>load_config_rc([conf_path])</code>	Load config from an rc-formatted file.
<code>reload_submodules(module_name)</code>	Helper function for reloading an existing module and its submodules.

##### example

`andes.system.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.fix_view_arrays(system)
```

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

**import\_pycode**

```
andes.system.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.

**load\_config\_rc**

```
andes.system.load_config_rc(conf_path=None)
```

Load config from an rc-formatted file.

**Parameters****conf\_path**

[`None` or `str`] Path to the config file. If is `None`, the function body will not run.

**Returns**

`configparse.ConfigParser`

## **reload\_submodules**

`andes.system.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**

<code>ExistingModels()</code>	Storage class for existing models
<code>System([case, name, config, config_path, ...])</code>	System contains models and routines for modeling and simulation.

### **andes.system.ExistingModels**

**class** `andes.system.ExistingModels`

Storage class for existing models

`__init__()`

#### **Methods**

### **andes.system.System**

**class** `andes.system.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.

## 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.

### Attributes

#### `dae`

[`andes.variables.dae.DAE`] Numerical DAE storage

#### `files`

[`andes.variables.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

**`__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

<code>add(model[, param_dict])</code>	Add a device instance for an existing model.
<code>as_dict([vin, skip_empty])</code>	Return system data as a dict where the keys are model names and values are dicts.
<code>calc_pu_coeff()</code>	Perform per unit value conversion.
<code>call_models(method, models, *args, **kwargs)</code>	Call methods on the given models.
<code>collect_config()</code>	Collect config data from models.
<code>collect_ref()</code>	Collect indices into <code>BackRef</code> for all models.
<code>connectivity([info])</code>	Perform connectivity check for system.
<code>e_clear(models)</code>	Clear equation arrays in DAE and model variables.
<code>f_update(models)</code>	Call the differential equation update method for models in sequence.
<code>fg_to_dae()</code>	Collect equation values into the DAE arrays.
<code>find_devices()</code>	Add dependent devices for all model based on <code>DeviceFinder</code> .

continues on next page

Table 1 – continued from previous page

<code>find_models(flag[, skip_zero])</code>	Find models with at least one of the flags as True.
<code>from_ipysheet(model, sheet[, vin])</code>	Set an ipysheet object back to model.
<code>g_islands()</code>	Reset algebraic mismatches for islanded buses.
<code>g_update(models)</code>	Call the algebraic equation update method for models in sequence.
<code>get_z(models)</code>	Get all discrete status flags in a numpy array.
<code>import_groups()</code>	Import all groups classes defined in <code>models/group.py</code> .
<code>import_models()</code>	Import and instantiate models as System member attributes.
<code>import_routines()</code>	Import routines as defined in <code>routines/__init__.py</code> .
<code>init(models, routine)</code>	Initialize the variables for each of the specified models.
<code>j_islands()</code>	Set gy diagonals to eps for $a$ and $v$ variables of islanded buses.
<code>j_update(models[, info])</code>	Call the Jacobian update method for models in sequence.
<code>l_update_eq(models[, init, niter])</code>	Update equation-dependent limiter discrete components by calling <code>l_check_eq</code> of models.
<code>l_update_var(models[, niter, err])</code>	Update variable-based limiter discrete states by calling <code>l_update_var</code> of models.
<code>link_ext_param([model])</code>	Retrieve values for <code>ExtParam</code> for the given models.
<code>precompile([models, nomp, ncpu])</code>	Trigger precompilation for the given models.
<code>prepare([quick, incremental, models, nomp, ncpu])</code>	Generate numerical functions from symbolically defined models.
<code>reload(case, **kwargs)</code>	Reload a new case in the same System object.
<code>remove_pycapsule()</code>	Remove PyCapsule objects in solvers.
<code>reset([force])</code>	Reset to the state after reading data and setup (before power flow).
<code>s_update_post(models)</code>	Update variable services by calling <code>s_update_post</code> of models.
<code>s_update_var(models)</code>	Update variable services by calling <code>s_update_var</code> of models.
<code>save_config([file_path, overwrite])</code>	Save all system, model, and routine configurations to an rc-formatted file.
<code>set_address(models)</code>	Set addresses for differential and algebraic variables.
<code>set_config([config])</code>	Set configuration for the System object.
<code>set_dae_names(models)</code>	Set variable names for differential and algebraic variables, right-hand side of external equations, and discrete flags.
<code>set_output_subidx(models)</code>	Process <code>andes.models.misc.Output</code> data and store the sub-indices into <code>dae.xy</code> .

continues on next page

Table 1 – continued from previous page

<code>set_var_arrays(models[, inplace, alloc])</code>	Set arrays ( $v$ and $e$ ) for internal variables to access dae arrays in place.
<code>setup()</code>	Set up system for studies.
<code>store_adder_setter(models)</code>	Store non-inplace adders and setters for variables and equations.
<code>store_existing()</code>	Store existing models in <i>System.existing</i> .
<code>store_no_check_init(models)</code>	Store differential variables with <code>check_init == False</code> .
<code>store_sparse_pattern(models)</code>	Collect and store the sparsity pattern of Jacobian matrices.
<code>store_switch_times(models[, eps])</code>	Store event switching time in a sorted Numpy array in <i>System.switch_times</i> and an OrderedDict <i>System.switch_dict</i> .
<code>summary()</code>	Print out system summary.
<code>supported_models([export])</code>	Return the support group names and model names in a table.
<code>switch_action(models)</code>	Invoke the actions associated with switch times.
<code>to_ipysheet(model[, vin])</code>	Return an ipysheet object for editing in Jupyter Notebook.
<code>undill([autogen_stale])</code>	Reload generated function functions, from either the \$HOME/.andes/pycode folder.
<code>vars_to_dae(model)</code>	Copy variables values from models to <i>System.dae</i> .
<code>vars_to_models()</code>	Copy variable values from <i>System.dae</i> to models.

## System.add

`System.add(model, param_dict=None, **kwargs)`

Add a device instance for an existing model.

This methods calls the `add` method of *model* and registers the device *idx* to group.

## System.as\_dict

`System.as_dict(vin=False, skip_empty=True)`

Return system data as a dict where the keys are model names and values are dicts. Each dict has parameter names as keys and corresponding data in an array as values.

### Returns

`OrderedDict`

## System.calc\_pu\_coeff

System.**calc\_pu\_coeff**()

Perform per unit value conversion.

This function calculates the per unit conversion factors, stores input parameters to *vin*, and perform the conversion.

## System.call\_models

System.**call\_models**(method: str, models: *OrderedDict*, \*args, \*\*kwargs)

Call methods on the given models.

### Parameters

#### method

[str] Name of the model method to be called

#### models

[OrderedDict, list, str] Models on which the method will be called

#### args

Positional arguments to be passed to the model method

#### kwargs

Keyword arguments to be passed to the model method

### Returns

The return value of the models in an *OrderedDict*

## System.collect\_config

System.**collect\_config**()

Collect config data from models.

### Returns

#### dict

a dict containing the config from devices; class names are keys and configs in a dict are values.

## System.collect\_ref

`System.collect_ref()`

Collect indices into *BackRef* for all models.

## System.connectivity

`System.connectivity(info=True)`

Perform connectivity check for system.

### Parameters

#### info

[bool] True to log connectivity summary.

## System.e\_clear

`System.e_clear(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.

## System.f\_update

`System.f_update(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.

## System.fg\_to\_dae

`System.fg_to_dae()`

Collect equation values into the DAE arrays.

Additionally, the function resets the differential equations associated with variables pegged by anti-windup limiters.

## System.find\_devices

System.**find\_devices** ()

Add dependent devices for all model based on *DeviceFinder*.

## System.find\_models

System.**find\_models** (flag: str | Tuple | None, skip\_zero: bool = True)

Find models with at least one of the flags as True.

### Parameters

#### flag

[list, str] Flags to find

#### skip\_zero

[bool] Skip models with zero devices

### Returns

#### OrderedDict

model name : model instance

**Warning:** Checking the number of devices has been centralized into this function. models passed to most System calls must be retrieved from here.

## System.from\_ipysheet

System.**from\_ipysheet** (model: str, sheet, vin: bool = False)

Set an ipysheet object back to model.

## System.g\_islands

System.**g\_islands** ()

Reset algebraic mismatches for islanded buses.

## System.g\_update

`System.g_update (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.

## System.get\_z

`System.get_z (models: OrderedDict)`

Get all discrete status flags in a numpy array. Values are written to `dae.z` in place.

### Returns

`numpy.array`

## System.import\_groups

`System.import_groups ()`

Import all groups classes defined in `models/group.py`.

Groups will be stored as instances with the name as class names. All groups will be stored to dictionary `System.groups`.

## System.import\_models

`System.import_models ()`

Import and instantiate models as System member attributes.

Models defined in `models/__init__.py` will be instantiated *sequentially* as attributes with the same name as the class name. In addition, all models will be stored in dictionary `System.models` with model names as keys and the corresponding instances as values.

## Examples

`system.Bus` stores the `Bus` object, and `system.GENCLS` stores the classical generator object, `system.models['Bus']` points the same instance as `system.Bus`.

## System.import\_routines

`System.import_routines()`

Import routines as defined in `routines/__init__.py`.

Routines will be stored as instances with the name as class names. All routines will be stored to dictionary `System.routines`.

## Examples

`System.PFlow` is the power flow routine instance, and `System.TDS` and `System.EIG` are time-domain analysis and eigenvalue analysis routines, respectively.

## System.init

`System.init(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.

## System.j\_islands

`System.j_islands()`

Set gy diagonals to eps for  $a$  and  $v$  variables of islanded buses.

## System.j\_update

`System.j_update(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).

### System.l\_update\_eq

`System.l_update_eq(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.

### System.l\_update\_var

`System.l_update_var(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.

### System.link\_ext\_param

`System.link_ext_param(model=None)`

Retrieve values for `ExtParam` for the given models.

### System.compile

`System.compile(models: OrderedDict | None = None, nomp: bool = False, ncpu: int = 1)`

Trigger precompilation for the given models.

Arguments are the same as `prepare`.

### System.prepare

`System.prepare(quick=False, incremental=False, models=None, nomp=False, ncpu=1)`

Generate numerical functions from symbolically defined models.

All procedures in this function must be independent of test case.

#### Parameters

##### quick

[bool, optional] True to skip pretty-print generation to reduce code generation time.

**incremental**

[bool, optional] True to generate only for modified models, incrementally.

**models**

[list, OrderedDict, None] List or OrderedDict of models to prepare

**nomp**

[bool] True to disable multiprocessing

**Warning:** Generated lambda functions will be serialized to file, but pretty prints (SymPy objects) can only exist in the System instance on which prepare is called.

**Notes**

Option `incremental` compares the md5 checksum of all var and service strings, and only regenerate for updated models.

**Examples**

If one needs to print out LaTeX-formatted equations in a Jupyter Notebook, one need to generate such equations with

```
import andes
sys = andes.prepare()
```

Alternatively, one can explicitly create a System and generate the code

```
import andes
sys = andes.System()
sys.prepare()
```

**System.reload**

`System.reload(case, **kwargs)`

Reload a new case in the same System object.

**System.remove\_pycapsule**

`System.remove_pycapsule()`

Remove PyCapsule objects in solvers.

## System.reset

`System.reset (force=False)`

Reset to the state after reading data and setup (before power flow).

**Warning:** If TDS is initialized, reset will lead to unpredictable state.

## System.s\_update\_post

`System.s_update_post (models: OrderedDict)`

Update variable services by calling `s_update_post` of models.

This function is called at the end of `System.init()`.

## System.s\_update\_var

`System.s_update_var (models: OrderedDict)`

Update variable services by calling `s_update_var` of models.

This function is must be called before any equation evaluation after limiter update function `l_update_var`.

## System.save\_config

`System.save_config (file_path=None, overwrite=False)`

Save all system, model, and routine configurations to an rc-formatted file.

### Parameters

#### file\_path

[str, optional] path to the configuration file default to `~/andes/andes.rc`.

#### overwrite

[bool, optional] If file exists, True to overwrite without confirmation. Otherwise prompt for confirmation.

**Warning:** Saved config is loaded back and populated *at system instance creation time*. Configs from the config file takes precedence over default config values.

## System.set\_address

System.**set\_address** (*models*)

Set addresses for differential and algebraic variables.

## System.set\_config

System.**set\_config** (*config=None*)

Set configuration for the System object.

Config for models are routines are passed directly to their constructors.

## System.set\_dae\_names

System.**set\_dae\_names** (*models*)

Set variable names for differential and algebraic variables, right-hand side of external equations, and discrete flags.

## System.set\_output\_subidx

System.**set\_output\_subidx** (*models*)

Process andes.models.misc.Output data and store the sub-indices into dae.xy.

### Parameters

#### models

[OrderedDict] Models currently in use for the routine

## System.set\_var\_arrays

System.**set\_var\_arrays** (*models, inplace=True, alloc=True*)

Set arrays (*v* and *e*) for internal variables to access dae arrays in place.

This function needs to be called after de-serializing a System object, where the internal variables are incorrectly assigned new memory.

### Parameters

#### models

[OrderedDict, list, Model, optional] Models to execute.

#### inplace

[bool] True to retrieve arrays that share memory with dae

#### alloc

[bool] True to allocate for arrays internally

## **System.setup**

`System.setup()`

Set up system for studies.

This function is to be called after adding all device data.

## **System.store\_adder.setter**

`System.store_adder.setter(models)`

Store non-inplace adders and setters for variables and equations.

## **System.store\_existing**

`System.store_existing()`

Store existing models in *System.existing*.

TODO: Models with *TimerParam* will need to be stored anyway. This will allow adding switches on the fly.

## **System.store\_no\_check\_init**

`System.store_no_check_init(models)`

Store differential variables with `check_init == False`.

## **System.store\_sparse\_pattern**

`System.store_sparse_pattern(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.

## System.store\_switch\_times

System.**store\_switch\_times** (*models*, *eps*=0.0001)

Store event switching time in a sorted Numpy array in System.switch\_times and an OrderedDict System.switch\_dict.

System.switch\_dict has keys as event times and values as the OrderedDict of model names and instances associated with the event.

### Parameters

#### models

[OrderedDict] model name : model instance

#### eps

[float] The small time step size to use immediately before and after the event

### Returns

#### array-like

self.switch\_times

## System.summary

System.**summary** ()

Print out system summary.

## System.supported\_models

System.**supported\_models** (*export*='plain')

Return the support group names and model names in a table.

### Returns

#### str

A table-formatted string for the groups and models

## System.switch\_action

System.**switch\_action** (*models*: *OrderedDict*)

Invoke the actions associated with switch times.

This function will not be called if `flat=True` is passed to system.

## System.to\_ipysheet

`System.to_ipysheet (model: str, vin: bool = False)`

Return an ipysheet object for editing in Jupyter Notebook.

## System.undill

`System.undill (autogen_stale=True)`

Reload generated function functions, from either the `$HOME/.andes/pycode` folder.

If no change is made to models, future calls to `prepare()` can be replaced with `undill()` for acceleration.

### Parameters

#### `autogen_stale: bool`

True to automatically call code generation if stale code is detected. Regardless of this option, codegen is trigger if importing existing code fails.

## System.vars\_to\_dae

`System.vars_to_dae (model)`

Copy variables values from models to `System.dae`.

This function clears `DAE.x` and `DAE.y` and collects values from models.

## System.vars\_to\_models

`System.vars_to_models ()`

Copy variable values from `System.dae` to models.

## 6.1.2 andes.variables

Variable package with classes for numerical DAE.

## Modules

---

`andes.variables.dae`

`andes.variables.fileman`

`andes.variables.report`

---

**andes.variables.dae****Classes**

<code>DAE(system)</code>	Class for storing numerical values of the DAE system, including variables, equations and first order derivatives (Jacobian matrices).
<code>DAETimeSeries([dae])</code>	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 algebraic 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**, , **gx**, and , 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

<code>alloc_or_extend_names()</code>	Allocate empty lists for names for the given size.
<code>build_pattern(name)</code>	Build sparse matrices with stored patterns.
<code>clear_arrays()</code>	Reset equation and variable arrays to empty.
<code>clear_fg()</code>	Resets equation arrays to empty.
<code>clear_ijv()</code>	Clear stored triplets.
<code>clear_ts()</code>	Drop the TimeSeries data and create a new one.
<code>clear_xy()</code>	Reset variable arrays to empty.
<code>clear_z()</code>	Reset status arrays to empty
<code>get_name(arr)</code>	Helper function for getting the list of variable names based on the array name.
<code>get_size(name)</code>	Get the size of an array or sparse matrix based on name.
<code>print_array(name[, values, tol])</code>	Debug helper to print array values and names.
<code>request_address(array_name, ndevice, nvar[, ...])</code>	Interface for requesting addresses for a model.
<code>reset()</code>	Reset array sizes to zero and clear all arrays.
<code>resize_arrays()</code>	Resize arrays to the new sizes $m$ and $n$ , and $o$ .
<code>restore_sparse([names])</code>	Restore all sparse matrices to the sparsity pattern filled with zeros (for variable Jacobian elements) and non-zero constants.
<code>set_t(t)</code>	Helper function for setting time in-place.
<code>store()</code>	Store values for the current time step to the Time-Series storage.
<code>store_sparse_ijv(name, row, col, val)</code>	Store the sparse pattern triplets.
<code>write_lst(lst_path)</code>	Dump the variable name lst file.
<code>write_npy(file_path)</code>	Write TDS data into NumPy uncompressed format.
<code>write_npz(file_path)</code>	Write TDS data into NumPy compressed format.

**DAE.alloc\_or\_extend\_names****DAE.alloc\_or\_extend\_names ()**

Allocate empty lists for names for the given size.

**DAE.build\_pattern****DAE.build\_pattern (name)**

Build sparse matrices with stored patterns.

Call to *store\_row\_col\_idx* should be made before this function.

**Parameters****name**

[name] jac name

**DAE.clear\_arrays****DAE.clear\_arrays ()**

Reset equation and variable arrays to empty.

**DAE.clear\_fg****DAE.clear\_fg ()**

Resets equation arrays to empty.

**DAE.clear\_ijv****DAE.clear\_ijv ()**

Clear stored triplets.

**DAE.clear\_ts****DAE.clear\_ts ()**

Drop the TimeSeries data and create a new one.

**DAE.clear\_xy****DAE . clear\_xy ()**

Reset variable arrays to empty.

**DAE.clear\_z****DAE . clear\_z ()**

Reset status arrays to empty

**DAE.get\_name****DAE . get\_name (arr)**

Helper function for getting the list of variable names based on the array name.

**Parameters****arr**

[str] Array name in 'f', 'g', 'x', 'y', 'z'.

**DAE.get\_size****DAE . get\_size (name)**

Get the size of an array or sparse matrix based on name.

**Parameters****name**

[str (f, g, fx, gy, etc.)] array/sparse name

**Returns****tuple**

sizes of each element in a tuple

**DAE.print\_array****DAE . print\_array (name, values=None, tol=None)**

Debug helper to print array values and names.

**Parameters****name**

[str] array name in 'f', 'g', 'x', 'y'

**values**

[array-like, optional] substitute array values to use

**tol**

[float, optional] tolerance value to use. Values below *tol* will not be displayed

**DAE.request\_address**

`DAE.request_address (array_name: str, ndevice, nvar, collate=False)`

Interface for requesting addresses for a model.

**Parameters****array\_name**

[str] array name in 'x' and 'y'

**ndevice**

[int] number of devices

**nvar**

[int] number of variables

**collate**

[bool, optional] False if the same variable for different devices are contiguous.  
True if variables for the same devices should collate. Note: setting `collate` to True will degrade the performance.

**Returns****list**

A list of arrays for each variable.

**DAE.reset**

`DAE.reset ()`

Reset array sizes to zero and clear all arrays.

**DAE.resize\_arrays**

`DAE.resize_arrays ()`

Resize arrays to the new sizes *m* and *n*, and *o*.

If *m* > `len(self.y)` or *n* > `len(self.x)`, arrays will be extended. Otherwise, new empty arrays will be sliced, starting from 0 to the given size.

**Warning:** This function should not be called directly. Instead, it is called in `System.set_address` which re-points variables used in power flow to the new array for dynamic analyses.

## DAE.restore\_sparse

DAE.**restore\_sparse** (*names=None*)

Restore all sparse matrices to the sparsity pattern filled with zeros (for variable Jacobian elements) and non-zero constants.

### Parameters

#### names

[None or list] List of Jacobian names to restore sparsity pattern

## DAE.set\_t

DAE.**set\_t** (*t*)

Helper function for setting time in-place.

## DAE.store

DAE.**store** ()

Store values for the current time step to the TimeSeries storage. Values include variables, equation RHS and discrete states.

## DAE.store\_sparse\_ijv

DAE.**store\_sparse\_ijv** (*name, row, col, val*)

Store the sparse pattern triplets.

This function is to be called by System after building the complete sparsity pattern for each Jacobian matrix.

### Parameters

#### name

[str] sparse Jacobian matrix name

#### row

[np.ndarray] all row indices

#### col

[np.ndarray] all col indices

#### val

[np.ndarray] all values

## DAE.write\_lst

DAE.**write\_lst** (*lst\_path*)

Dump the variable name lst file.

### Parameters

#### lst\_path

Path to the lst file.

### Returns

#### bool

succeed flag

## DAE.write\_npy

DAE.**write\_npy** (*file\_path*)

Write TDS data into NumPy uncompressed format.

## DAE.write\_npz

DAE.**write\_npz** (*file\_path*)

Write TDS data into NumPy compressed format.

The function supports writing out all values at once or writing them out incrementally.

## Attributes

<code>fg</code>	Return a concatenated array of [f, g].
<code>x_name_output</code>	Return a list of state var names selected by Output.
<code>x_tex_name_output</code>	Return a list of state var LaTeX names selected by Output.
<code>xy</code>	Return a concatenated array of [x, y].
<code>xy_name</code>	Return a concatenated list of all variable names without format.
<code>xy_tex_name</code>	Return a concatenated list of all variable names in LaTeX format.
<code>xyz</code>	Return a concatenated array of [x, y].
<code>xyz_name</code>	Return a concatenated list of all variable names without format.
<code>xyz_tex_name</code>	Return a concatenated list of all variable names in LaTeX format.
<code>y_name_output</code>	Return a list of algeb var names selected by Output.
<code>y_tex_name_output</code>	Return a list of algeb var LaTeX names selected by Output.

### DAE.fg

**property DAE.fg**

Return a concatenated array of [f, g].

### DAE.x\_name\_output

**property DAE.x\_name\_output**

Return a list of state var names selected by Output.

### DAE.x\_tex\_name\_output

**property DAE.x\_tex\_name\_output**

Return a list of state var LaTeX names selected by Output.

**DAE.xy****property DAE.xy**

Return a concatenated array of [x, y].

**DAE.xy\_name****property DAE.xy\_name**

Return a concatenated list of all variable names without format.

**DAE.xy\_tex\_name****property DAE.xy\_tex\_name**

Return a concatenated list of all variable names in LaTeX format.

**DAE.xyz****property DAE.xyz**

Return a concatenated array of [x, y].

**DAE.xyz\_name****property DAE.xyz\_name**

Return a concatenated list of all variable names without format.

**DAE.xyz\_tex\_name****property DAE.xyz\_tex\_name**

Return a concatenated list of all variable names in LaTeX format.

**DAE.y\_name\_output****property DAE.y\_name\_output**

Return a list of algeb var names selected by Output.

## DAE.y\_tex\_name\_output

**property DAE.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

<code>get_data(base_vars, *[, a, rhs])</code>	Get time-series data, either for a variable or for the equation associated with the variable.
<code>reset()</code>	Reset the internal storage and erase all data.
<code>unpack([df, attr, warn_empty])</code>	Unpack dict-stored data into arrays and/or dataframes.
<code>unpack_df(attr)</code>	Construct pandas dataframes.
<code>unpack_np(attr[, warn_empty])</code>	Unpack dict data into numpy arrays.

### DAETimeSeries.get\_data

`DAETimeSeries.get_data (base_vars: BaseVar | List[BaseVar], *, a=None, rhs: bool = False)`

Get time-series data, either for a variable or for the equation associated with the variable.

#### Parameters

##### **base\_var**

[BaseVar or a sequence of BaseVar(s)] The variable types and internal addresses are used for looking up the data.

##### **a**

[an array/list of int or None] Sub-indices into the address of *base\_var*. Applied to each variable.

#### Returns

##### **np.ndarray**

A two-dimensional array. Each row corresponds to one time step. Each column corresponds to a different different variable.

## DAETimeSeries.reset

DAETimeSeries.**reset**()

Reset the internal storage and erase all data.

## DAETimeSeries.unpack

DAETimeSeries.**unpack**(*df=False*, *attr=None*, *warn\_empty=True*)

Unpack dict-stored data into arrays and/or dataframes.

### Parameters

#### df

[bool] True to construct DataFrames *self.df* and *self.df\_z* (time-consuming).

#### attr

[str, optional] Attribute name to unpack. If None, unpack all.

### Returns

**True when done.**

## DAETimeSeries.unpack\_df

DAETimeSeries.**unpack\_df**(*attr*)

Construct pandas dataframes.

## DAETimeSeries.unpack\_np

DAETimeSeries.**unpack\_np**(*attr*, *warn\_empty=True*)

Unpack dict data into numpy arrays.

## Attributes

---

*df*

Short-hand for the xy dataframe.

---

## DAETimeSeries.df

**property** DAETimeSeries.**df**

Short-hand for the xy dataframe.

## andes.variables.fileman

### Functions

`add_suffix`(fullname, suffix)

Add suffix to a full file name.

### add\_suffix

`andes.variables.fileman.add_suffix`(*fullname*, *suffix*)

Add suffix to a full file name.

### 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

<code>get_fullpath([fullname])</code>	Return the original full path if full path is specified, otherwise search in the case file path.
<code>set([case])</code>	Perform the input and output set up.

### FileMan.get\_fullpath

`FileMan.get_fullpath(fullname=None)`

Return the original full path if full path is specified, otherwise search in the case file path.

#### Parameters

##### fullname

[str, optional] Full name of the file. If relative, prepend *input\_path*. Otherwise, leave it as is.

### FileMan.set

`FileMan.set(case=None, **kwargs)`

Perform the input and output set up.

### andes.variables.report

## Functions

---

`report_info(system)`

---

### report\_info

`andes.variables.report.report_info(system)`

## Classes

<code>Report(system)</code>	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

<code>update()</code>	Update values based on the requested content
<code>write()</code>	Write report to file.

#### **Report.update**

`Report.update()`

Update values based on the requested content

#### **Report.write**

`Report.write()`

Write report to file.

#### Attributes

`info`

## Report.info

```
property Report.info
```

## 6.2 Routines

*andes.routines*

Package for ANDES analysis routines.

### 6.2.1 andes.routines

Package for ANDES analysis routines.

#### Modules

<i>andes.routines.base</i>	Base class for ANDES calculation routines.
<i>andes.routines.criteria</i>	Stability criteria module.
<i>andes.routines.daeint</i>	Integration methods for DAE.
<i>andes.routines.eig</i>	Module for eigenvalue analysis.
<i>andes.routines.pflow</i>	Module for power flow calculation.
<i>andes.routines.tds</i>	ANDES module for time-domain simulation.

#### andes.routines.base

Base class for ANDES calculation routines.

#### Classes

<i>BaseRoutine</i> ([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

<code>doc([max_width, export])</code>	Routine documentation interface.
<code>init()</code>	Routine initialization interface.
<code>report(**kwargs)</code>	Report interface.
<code>run(**kwargs)</code>	Routine main entry point.
<code>summary(**kwargs)</code>	Summary interface

### **BaseRoutine.doc**

`BaseRoutine.doc (max_width=78, export='plain')`

Routine documentation interface.

### **BaseRoutine.init**

`BaseRoutine.init ()`

Routine initialization interface.

### **BaseRoutine.report**

`BaseRoutine.report (**kwargs)`

Report interface.

### **BaseRoutine.run**

`BaseRoutine.run (**kwargs)`

Routine main entry point.

### **BaseRoutine.summary**

`BaseRoutine.summary (**kwargs)`

Summary interface

## Attributes

*class\_name*

### BaseRoutine.class\_name

**property** BaseRoutine.**class\_name**

## 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.

## andes.routines.daeint.BackEuler

```
class andes.routines.daeint.BackEuler
```

Backward Euler's integration method.

```
__init__(*args, **kwargs)
```

### Methods

<code>calc_jac(tds, gxs, gys)</code>	Build full Jacobian matrix $A_C$ for Trapezoid method.
<code>calc_q(x, f, Tf, h, x0, f0)</code>	Calculate the residual of algebraized differential equations.
<code>step(tds)</code>	Integrate with Implicit Trapezoidal Method (ITM) to the current time.

#### BackEuler.calc\_jac

```
static BackEuler.calc_jac(tds, gxs, gys)
```

Build full Jacobian matrix  $A_C$  for Trapezoid method.

#### BackEuler.calc\_q

```
static BackEuler.calc_q(x, f, Tf, h, x0, f0)
```

Calculate the residual of algebraized differential equations.

### Notes

Numba jit somehow slows down this function for the 14-bus and the 2k-bus systems.

#### BackEuler.step

```
static BackEuler.step(tds)
```

Integrate with Implicit Trapezoidal Method (ITM) to the current time.

This function has an internal Newton-Raphson loop for algebraized semi-explicit DAE. The function returns the convergence status when done but does NOT progress simulation time.

### Returns

`bool`

Convergence status in `tds.converged`.

**andes.routines.daeint.ImplicitIter**

```
class andes.routines.daeint.ImplicitIter
```

Base class for implicit iterative methods.

```
__init__(*args, **kwargs)
```

**Methods**

<i>calc_jac</i> (tds, gxs, gys)
---------------------------------

<i>calc_q</i> (x, f, Tf, h, x0, f0)
-------------------------------------

<i>step</i> (tds)
-------------------

	Integrate with Implicit Trapezoidal Method (ITM) to the current time.
--	--

**ImplicitIter.calc\_jac**

```
static ImplicitIter.calc_jac(tds, gxs, gys)
```

**ImplicitIter.calc\_q**

```
static ImplicitIter.calc_q(x, f, Tf, h, x0, f0)
```

**ImplicitIter.step**

```
static ImplicitIter.step(tds)
```

Integrate with Implicit Trapezoidal Method (ITM) to the current time.

This function has an internal Newton-Raphson loop for algebraized semi-explicit DAE. The function returns the convergence status when done but does NOT progress simulation time.

**Returns**

**bool**

Convergence status in `tds.converged`.

## andes.routines.daeint.Trapezoid

```
class andes.routines.daeint.Trapezoid
```

Trapezoidal methods.

```
__init__(*args, **kwargs)
```

### Methods

<code>calc_jac(tds, gxs, gys)</code>	Build full Jacobian matrix $A_C$ for Trapezoid method.
<code>calc_q(x, f, Tf, h, x0, f0)</code>	Calculate the residual of algebraized differential equations.
<code>step(tds)</code>	Integrate with Implicit Trapezoidal Method (ITM) to the current time.

### Trapezoid.calc\_jac

```
static Trapezoid.calc_jac(tds, gxs, gys)
```

Build full Jacobian matrix  $A_C$  for Trapezoid method.

### Trapezoid.calc\_q

```
static Trapezoid.calc_q(x, f, Tf, h, x0, f0)
```

Calculate the residual of algebraized differential equations.

### Notes

Numba jit somehow slows down this function for the 14-bus and the 2k-bus systems.

### Trapezoid.step

```
static Trapezoid.step(tds)
```

Integrate with Implicit Trapezoidal Method (ITM) to the current time.

This function has an internal Newton-Raphson loop for algebraized semi-explicit DAE. The function returns the convergence status when done but does NOT progress simulation time.

### Returns

`bool`

Convergence status in `tds.converged`.

**andes.routines.eig**

Module for eigenvalue analysis.

**Classes**

<code>EIG(system, config)</code>	Eigenvalue analysis routine
----------------------------------	-----------------------------

**andes.routines.eig.EIG**

```
class andes.routines.eig.EIG(system, config)
Eigenvalue analysis routine
__init__(system, config)
```

**Methods**

<code>calc_As([dense])</code>	Return state matrix and store to <code>self.As</code> .
<code>calc_eig([As])</code>	Calculate eigenvalues and right eigen vectors.
<code>calc_pfactor([As])</code>	Compute participation factor of states in eigenvalues.
<code>doc([max_width, export])</code>	Routine documentation interface.
<code>export_mat()</code>	Export state matrix to a <CaseName>.As.mat file with the variable name As, where <CaseName> is the test case name.
<code>find_zero_states()</code>	Find the indices of states associated with zero time constants in x.
<code>init()</code>	Routine initialization interface.
<code>plot([mu, fig, ax, left, right, ymin, ymax, ...])</code>	Plot utility for eigenvalues in the S domain.
<code>plot_root_loci(results, eig_indices[, ax, ...])</code>	Plot the root loci.
<code>post_process()</code>	Post processing of eigenvalues.
<code>report([x_name])</code>	Save eigenvalue analysis reports.
<code>run(**kwargs)</code>	Run small-signal stability analysis.
<code>stats()</code>	Return statistics of results in a string.
<code>summary()</code>	Print out a summary to <code>logger.info</code> .
<code>sweep(params, idxes, values)</code>	Parameter sweep for root loci plot.

## EIG.calc\_As

`EIG.calc_As (dense=True)`

Return state matrix and store to `self.As`.

### Returns

`kvxopt.matrix`

state matrix

## Notes

For systems in the mass-matrix formulation,

$$\begin{aligned} T\dot{x} &= f(x, y) \\ 0 &= g(x, y) \end{aligned}$$

Assume  $T$  is non-singular, the state matrix is calculated from

$$A_s = T^{-1}(f_x - f_y * g_y^{-1} * g_x)$$

## EIG.calc\_eig

`EIG.calc_eig (As=None)`

Calculate eigenvalues and right eigen vectors.

This function is a wrapper to `np.linalg.eig`. Results are returned but not stored to `EIG`.

### Returns

`np.array(dtype=complex)`

eigenvalues

`np.array()`

right eigenvectors

## EIG.calc\_pfactor

`EIG.calc_pfactor (As=None)`

Compute participation factor of states in eigenvalues.

Each row in the participation factor correspond to one state, and each column correspond to one mode.

### Parameters

`As`

[`np.array` or `None`] State matrix to process. If `None`, use `self.As`.

### Returns

```
np.array(dtype=complex)
eigenvalues

np.array
participation factor matrix
```

## EIG.doc

**EIG.doc** (*max\_width*=78, *export*='plain')

Routine documentation interface.

## EIG.export\_mat

**EIG.export\_mat()**

Export state matrix to a <CaseName>\_As.mat file with the variable name As, where <CaseName> is the test case name.

State variable names are stored in variables x\_name and x\_tex\_name.

### Returns

**bool**

True if successful

## EIG.find\_zero\_states

**EIG.find\_zero\_states()**

Find the indices of states associated with zero time constants in x.

## EIG.init

**EIG.init()**

Routine initialization interface.

## EIG.plot

**EIG.plot** (*mu*=None, *fig*=None, *ax*=None, *left*=-6, *right*=0.5, *ymin*=-8, *ymax*=8, *damping*=0.05,  
*line\_width*=0.5, *s*=40, *dpi*=None, *figsize*=None, *base\_color*='black', *show*=True,  
*latex*=True, *style*='default')

Plot utility for eigenvalues in the S domain.

### Parameters

**mu**

[array, optional] an array of complex eigenvalues

**fig**

[figure handle, optional] existing matplotlib figure handle

**ax**

[axis handle, optional] existing axis handle

**left**

[int, optional] left tick for the x-axis, by default -6

**right**

[float, optional] right tick, by default 0.5

 **ymin**

[int, optional] bottom tick, by default -8

 **ymax**

[int, optional] top tick, by default 8

**damping**

[float, optional] damping value for which the dash plots are drawn

**line\_width**

[float, optional] default line width, by default 0.5

**s**

[float or array-like, shape (n, ), optional] The marker size in points\*\*2

**dpi**

[int, optional] figure dpi

 **figsize**

[[type], optional] default figure size, by default None

**base\_color**

[str, optional] base color for negative eigenvalues

 **show**

[bool, optional] True to show figure after plot, by default True

 **latex**

[bool, optional] True to use latex, by default True

**Returns****figure**

matplotlib figure object

**axis**

matplotlib axis object

## EIG.plot\_root\_loci

```
EIG.plot_root_loci(results, eig_indices, ax=None, dpi=None, figsize=None, draw_line=False, arrow_threshold=0.2, **kwargs)
```

Plot the root loci.

Markers increase in size for the first parameter through the last.

### Parameters

**results**

[dict] Eigenvalue results from parameter sweeping

**eig\_indices**

[Iterable] A list of eigenvalue indices to plot. The indices are 0-based, whereas the indices in the eigenvalue analysis report are 1-based.

**ax**

[matplotlib.axes.Axes or None] Axes to plot on. If None, create a new figure.

**dpi**

[int or None] DPI of the figure. If None, use the default DPI.

**figsize**

[tuple or None] Figure size. If None, use the default size.

**draw\_line**

[bool, optional, False by default] If True, draw lines to connect the roots. Note that due to the non-fixed ordering of eigenvalues, lines will largely connect different modes!

**arrow\_threshold**

[float] Threshold for plotting arrows. If the begin and end points of a locus is shorter than this threshold, no arrow is plotted.

### Returns

**matplotlib.figure.Figure**

Figure containing the plot.

**matplotlib.axes.Axes**

Axes of the plot.

## Examples

To plot the root loci of the first two eigenvalues, do

```
fig, ax = ss.EIG.plot_root_loci(ret, [0, 1])
```

where `ret` is the return of `andes.routines.eig.EIG.sweep()`.

**EIG.post\_process****EIG.post\_process ()**

Post processing of eigenvalues.

**EIG.report****EIG.report (x\_name=None, \*\*kwargs)**

Save eigenvalue analysis reports.

**Returns****None****EIG.run****EIG.run (\*\*kwargs)**

Run small-signal stability analysis.

**EIG.stats****EIG.stats ()**

Return statistics of results in a string.

**EIG.summary****EIG.summary ()**

Print out a summary to logger.info.

**EIG.sweep****EIG.sweep (params, idxes, values)**

Parameter sweep for root loci plot.

**Parameters****params**

[list of NumParam or ConstService] list of parameters indices to sweep. For example, [ss.GENCLS.M] for GENCLS.M. To update ss.GENCLS.M for two generators, params should be set to [ss.GENCLS.M, ss.GENCLS.M].

**idxes**

[list of int or str] list of indices to sweep. For example, [ "GENCLS\_1" , "GENCLS\_2" ] for the indices of GENCLS whose corresponding parameter will be updated. The length of `idxes` must match that of `params` and `values`.

**values: list of lists**

New values of the parameters. Each element in `values` is a list for the corresponding element in `params` and `idxes`.

**Returns****dict**

A dictionary of the results where the keys are 0-indexed count of parameter set, and the values are dictionaries. Each value dictionary contains a `mu` field for the eigenvalues.

**Examples**

To apply 10 parameters evenly spaced between 1 and 10 to `ss.GENCLS.M` of `GENCLS_1`, do

```
ret = ss.EIG.sweep(ss.GENCLS.M, "GENCLS_1", np.linspace(1, 2, 10))
```

This is equivalent to the following just for convenience.

```
ret = ss.EIG.sweep([ss.GENCLS.M],
                   ["GENCLS_1"],
                   [np.linspace(1, 2, 10)])
```

**Attributes**


---

`class_name`

---

**EIG.class\_name**

```
property EIG.class_name
```

## andes.routines.pflow

Module for power flow calculation.

### Classes

<code>PFlow([system, config])</code>	Power flow calculation routine.
--------------------------------------	---------------------------------

## andes.routines.pflow.PFlow

**class** andes.routines.pflow.**PFlow**(*system=None, config=None*)

Power flow calculation routine.

**\_\_init\_\_**(*system=None, config=None*)

### Methods

<code>doc([max_width, export])</code>	Routine documentation interface.
<code>fg_update()</code>	Evaluate the limiters and residual equations.
<code>init()</code>	Initialize variables for power flow.
<code>newton_krylov([verbose])</code>	Full Newton-Krylov method from SciPy.
<code>nr_solve()</code>	Solve the power flow problem using iterative Newton's method.
<code>nr_step()</code>	Solve a single iteration step using the Newton-Raphson method.
<code>report()</code>	Write power flow report to a plain-text file.
<code>run(**kwargs)</code>	Solve the power flow using the selected method.
<code>summary()</code>	Output a summary for the PFlow routine.

### PFlow.doc

`PFlow.doc(max_width=78, export='plain')`

Routine documentation interface.

## PFlow.fg\_update

```
PFlow.fg_update()
```

Evaluate the limiters and residual equations.

## PFlow.init

```
PFlow.init()
```

Initialize variables for power flow.

## PFlow.newton\_krylov

```
PFlow.newton_krylov(verbose=True)
```

Full Newton-Krylov method from SciPy.

### Parameters

**verbose**

True if verbose.

### Returns

**bool**

Convergence status

**Warning:** The result might be wrong if discrete are in use!

## PFlow.nr\_solve

```
PFlow.nr_solve()
```

Solve the power flow problem using iterative Newton's method.

## PFlow.nr\_step

```
PFlow.nr_step()
```

Solve a single iteration step using the Newton-Raphson method.

### Returns

**float**

maximum absolute mismatch

## PFlow.report

`PFlow.report()`

Write power flow report to a plain-text file.

### Returns

`bool`

True if report was written, False otherwise.

## PFlow.run

`PFlow.run(**kwargs)`

Solve the power flow using the selected method.

### Returns

`bool`

convergence status

## PFlow.summary

`PFlow.summary()`

Output a summary for the PFlow routine.

## Attributes

---

`class_name`

---

## PFlow.class\_name

`property PFlow.class_name`

## andes.routines.tds

ANDES module for time-domain simulation.

## Classes

<code>TDS([system, config])</code>	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

<code>calc_h([resume])</code>	Calculate the time step size during the TDS.
<code>check_criteria()</code>	Check stability criteria.
<code>do_switch()</code>	Checks if is an event time and perform switch if true.
<code>doc([max_width, export])</code>	Routine documentation interface.
<code>fg_update(models[, init])</code>	Perform one round of evaluation for one iteration step.
<code>init()</code>	Initialize the status, storage and values for TDS.
<code>init_resume()</code>	Initialize a resumed simulation.
<code>itm_step()</code>	Integrate for the step size of <code>self.h</code> using implicit trapezoid method.
<code>load_plotter()</code>	Manually load a plotter into <code>TDS.plotter</code> .
<code>report(**kwargs)</code>	Report interface.
<code>reset()</code>	Reset internal states to pre-init condition.
<code>rewind(t)</code>	TODO: rewind to a past time.
<code>run([no_summary])</code>	Run time-domain simulation using numerical integration.
<code>save_output([npz])</code>	Save the simulation data into two files: a <code>.lst</code> file and a <code>.npz</code> file.
<code>set_method([name])</code>	Set DAE solution method.
<code>streaming_init()</code>	Send out initialization variables and process init from modules.
<code>streaming_step()</code>	Sync, handle and streaming for each integration step.
<code>summary()</code>	Print out a summary of TDS options to <code>logger.info</code> .
<code>test_init()</code>	Test if the TDS initialization is successful.

## TDS.calc\_h

`TDS.calc_h(resume=False)`

Calculate the time step size during the TDS.

### Parameters

#### resume

[bool] If True, calculate the initial step size.

### Returns

#### float

computed time step size stored in `self.h`

## Notes

A heuristic function is used for variable time step size

```
    min(0.50 * h, hmin), if niter >= 15  
h =  max(1.10 * h, hmax), if niter <= 6  
    min(0.95 * h, hmin), otherwise
```

## TDS.check\_criteria

`TDS.check_criteria()`

Check stability criteria.

## TDS.do\_switch

`TDS.do_switch()`

Checks if is an event time and perform switch if true.

## TDS.doc

`TDS.doc(max_width=78, export='plain')`

Routine documentation interface.

## TDS.fg\_update

`TDS.fg_update (models, init=False)`

Perform one round of evaluation for one iteration step. The following operations are performed in order:

- variable service updating through `s_update_var`
- discrete flags updating through `l_update_var`
- evaluation of the right-hand-side of  $f$
- equation-dependent discrete flags updating through `l_update_eq`
- evaluation of the right-hand-side of  $g$
- collection of residuals into dae through `fg_to_dae`.

## TDS.init

`TDS.init ()`

Initialize the status, storage and values for TDS.

### Returns

**array-like**

The initial values of  $xy$ .

## TDS.init\_resume

`TDS.init_resume ()`

Initialize a resumed simulation.

## TDS.item\_step

`TDS.item_step ()`

Integrate for the step size of `self.h` using implicit trapezoid method.

### Returns

**bool**

Convergence status in `self.converged`.

**TDS.load\_plotter**

```
TDS.load_plotter()
```

Manually load a plotter into TDS.plotter.

**TDS.report**

```
TDS.report(**kwargs)
```

Report interface.

**TDS.reset**

```
TDS.reset()
```

Reset internal states to pre-init condition.

**TDS.rewind**

```
TDS.rewind(t)
```

TODO: rewind to a past time.

**TDS.run**

```
TDS.run(no_summary=False, **kwargs)
```

Run time-domain simulation using numerical integration.

The default method is the Implicit Trapezoidal Method (ITM).

**TDS.save\_output**

```
TDS.save_output(npz=True)
```

Save the simulation data into two files: a .lst file and a .npz file.

This function saves the output regardless of the *files.no\_output* flag.

**Parameters****npz**

[bool] True to save in npz format; False to save in npy format.

**Returns****bool**

True if files are written. False otherwise.

## TDS.set\_method

`TDS.set_method(name: str = 'trapezoid')`

Set DAE solution method.

### Parameters

#### name

[str, optional, default: 'trapezoid'] DAE solver name

## TDS.streaming\_init

`TDS.streaming_init()`

Send out initialization variables and process init from modules.

### Returns

`None`

## TDS.streaming\_step

`TDS.streaming_step()`

Sync, handle and streaming for each integration step.

### Returns

`None`

## TDS.summary

`TDS.summary()`

Print out a summary of TDS options to logger.info.

### Returns

`None`

## TDS.test\_init

`TDS.test_init()`

Test if the TDS initialization is successful.

This function update `dae.f` and `dae.g` and checks if the residuals are zeros.

## Attributes

```
class_name
```

### TDS.class\_name

```
property TDS.class_name
```

## 6.3 Plot

---

```
andes.plot
```

The ANDES plotting tool.

---

### 6.3.1 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] <= upper.

#### 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=(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

<code>TDSData([full_name, mode, dae, path])</code>	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

<code>bqplot_data(xdata, ydata, *[xheader, ...])</code>	Plot with bqplot.
<code>data_to_df()</code>	Convert to pandas.DataFrame
<code>export_csv([path, idx, header, formatted, ...])</code>	Export to a csv file.
<code>find(query[, exclude, formatted, idx_only])</code>	Return variable names and indices matching <i>query</i> .
<code>get_call([backend])</code>	Get the internal <i>plot_data</i> function for the specified backend.
<code>get_header(idx[, formatted])</code>	Return a list of the variable names at the given indices.
<code>get_values(idx)</code>	Return the variable values at the given indices.
<code>guess_event_time()</code>	Guess the event starting time from the input data by checking when the values start to change
<code>load_dae()</code>	Load from DAE time series.
<code>load_lst()</code>	Load the lst file into internal data structures <i>_idx</i> , <i>_fname</i> , <i>_uname</i> , and counts the number of variables to <i>nvars</i> .
<code>load_npy_or_csv([delimiter])</code>	Load the npy, zpy or (the legacy) csv file into the internal data structure <i>self.xy</i> .
<code>panoview(mdl, vars, *[ncols, idx, a, figsize])</code>	Panoramic view of variables of a given model instance.
<code>plot(yidx[, xidx, a, ytimes, ycalc, left, ...])</code>	Entry function for plotting.
<code>plot_data(xdata, ydata, *[xheader, ...])</code>	Plot lines for the supplied data and options.
<code>plotn(nrows, ncols, yidxes[, xidxes, dpi, ...])</code>	Plot multiple subfigures in one figure.

## **TDSData.bqplot\_data**

```
TDSData.bqplot_data(xdata, ydata, *, xheader=None, yheader=None,  
    xlabel=None, ylabel=None, left=None, right=None, ymin=None, ymax=None,  
    legend=True, grid=False, fig=None, dpi=None, line_width=1.0,  
    greyscale=False, savefig=None, save_format=None, title=None,  
    **kargs)
```

Plot with bqplot. Experimental and incomplete.

## **TDSData.data\_to\_df**

```
TDSData.data_to_df()
```

Convert to pandas.DataFrame

## **TDSData.export\_csv**

```
TDSData.export_csv(path=None, idx=None, header=None, formatted=False, sort_idx=True,  
    fmt='%.18e')
```

Export to a csv file.

### **Parameters**

#### **path**

[str] path of the csv file to save

#### **idx**

[None or array-like, optional] the indices of the variables to export. Export all by default

#### **header**

[None or array-like, optional] customized header if not *None*. Use the names from the lst file by default

#### **formatted**

[bool, optional] Use LaTeX-formatted header. Does not apply when using customized header

#### **sort\_idx**

[bool, optional] Sort by idx or not, # TODO: implement sort

#### **fmt**

[str] cell formatter

### **Returns**

#### **str**

The path of the exported csv file

## TDSData.find

TDSData.**find**(*query*, *exclude=None*, *formatted=False*, *idx\_only=False*)

Return variable names and indices matching *query*.

### Parameters

#### **query**

[str] The string for querying variables. Multiple conditions can be separated by comma without space.

#### **exclude**

[str, optional] A string pattern to be excluded

#### **formatted**

[bool, optional] True to return formatted names, False otherwise

#### **idx\_only**

[bool, optional] True if only return indices

### Returns

#### **(list, list)**

(List of found indices, list of found names)

## TDSData.get\_call

TDSData.**get\_call**(*backend=None*)

Get the internal *plot\_data* function for the specified backend.

## TDSData.get\_header

TDSData.**get\_header**(*idx*, *formatted=False*)

Return a list of the variable names at the given indices.

### Parameters

#### **idx**

[list or int] The indices of the variables to retrieve

#### **formatted**

[bool] True to retrieve latex-formatted names, False for unformatted names

### Returns

#### **list**

A list of variable names (headers)

## **TDSData.get\_values**

`TDSData.get_values(idx)`

Return the variable values at the given indices.

### **Parameters**

#### **idx**

[list] The indicex of the variables to retrieve.  $idx=0$  is for Time. Variable indices start at 1.

### **Returns**

#### **np.ndarray**

Variable data

## **TDSData.guess\_event\_time**

`TDSData.guess_event_time()`

Guess the event starting time from the input data by checking when the values start to change

## **TDSData.load\_dae**

`TDSData.load_dae()`

Load from DAE time series.

## **TDSData.load\_lst**

`TDSData.load_lst()`

Load the lst file into internal data structures  $_idx$ ,  $_fname$ ,  $_uname$ , and counts the number of variables to  $nvars$ .

### **Returns**

#### **None**

## **TDSData.load\_npy\_or\_csv**

`TDSData.load_npy_or_csv(delimiter=',')`

Load the npy, zpy or (the legacy) csv file into the internal data structure `self._xy`.

### **Parameters**

#### **delimiter**

[str, optional] The delimiter for the case file. Default to comma.

### **Returns**

**None****TDSData.panoview**

```
TDSData.panoview(mdl, vars, *, ncols=3, idx=None, a=None, figsize=None, **kwargs)
```

Panoramic view of variables of a given model instance.

Select variables through `vars`. Select devices through `idx` or `a`, which has a higher priority.

This function also takes other arguments recognizable by `self.plot`.

**Parameters****mdl**

[ModelBase] Model instance

**var**

[list of str] A list of variable names to display

**ncol**

[int] Number of columns

**idx**

[list] A list of device idx-es for showing

**a**

[list of int] A list of device 0-based positions for showing

**figsize**

[tuple] Figure size for plotting

**Examples**

To plot `omega` and `delta` of GENROUs `GENROU_1` and `GENROU_2`:

```
system.TDS.plt.plot(system.GENROU,
                     vars=['omega', 'delta'],
                     idx=['GENROU_1', 'GENROU_2'])
```

**TDSData.plot**

```
TDSData.plot(yidx, xidx=(0,), *, a=None, ytimes=None, ycalc=None, left=None, right=None,
              ymin=None, ymax=None, xlabel=None, ylabel=None, xheader=None,
              yheader=None, legend=None, grid=False, greyscale=False, latex=True, dpi=None,
              line_width=1.0, font_size=12, savefig=None, save_format=None, show=True,
              title=None, linestyles=None, use_bqplot=False, hline1=None, hline2=None,
              vline1=None, vline2=None, hline=None, vline=None, fig=None, ax=None,
              backend=None, set_xlim=True, set_ylim=True, autoscale=False,
              legend_bbox=None, legend_loc=None, legend_ncol=1, figsize=None, color=None,
              **kwargs)
```

Entry function for plotting.

This function retrieves the x and y values based on the *xidx* and *yidx* inputs, applies scaling functions *ytimes* and *ycalc* sequentially, and delegates the plotting to the backend.

### Parameters

#### **yidx**

[list or int] The indices for the y-axis variables

#### **xidx**

[tuple or int, optional] The index for the x-axis variable

#### **a**

[tuple or list, optional] The 0-indexed sub-indices into *yidx* to plot.

#### **ytimes**

[float, optional] A scaling factor to apply to all y values.

#### **left**

[float] The starting value of the x axis

#### **right**

[float] The ending value of the x axis

#### **ymin**

[float] The minimum value of the y axis

#### **ymax**

[float] The maximum value of the y axis

#### **ylabel**

[str] Text label for the y axis

#### **yheader**

[list] A list containing the variable names for the y-axis variable

#### **title**

[str] Title string to be shown at the top

#### **fig**

Existing figure object to draw the axis on.

#### **ax**

Existing axis object to draw the lines on.

### Returns

#### **(fig, ax)**

Figure and axis handles for matplotlib backend.

#### **fig**

Figure object for bqplot backend.

### Other Parameters

**ycalc: callable, optional**

A callable to apply to all y values after scaling with *ytimes*.

**xlabel**

[str] Text label for the x axis

**xheader**

[list] A list containing the variable names for the x-axis variable

**legend**

[bool] True to show legend and False otherwise

**legend\_ncol**

[int] Number of columns in legend

**legend\_bbox**

[tuple of two floats] legend box to anchor

**grid**

[bool] True to show grid and False otherwise

**latex**

[bool] True to enable latex and False to disable

**greyscale**

[bool] True to use greyscale, False otherwise

**savefig**

[bool or str] True to save to png figure file. str is treated as the output file name.

**save\_format**

[str] File extension string (pdf, png or jpg) for the savefig format

**dpi**

[int] Dots per inch for screen print or save. *savefig* uses a minimum of 200 dpi

**line\_width**

[float] Plot line width

**font\_size**

[float] Text font size (labels and legends)

**figsize**

[tuple] Figure size passed when creating new figure

**show**

[bool] True to show the image

**backend**

[str or None] *bqplot* to use the bqplot backend in notebook. None for matplotlib.

**hline1: float, optional**

Dashed horizontal line 1

**hline2: float, optional**

Dashed horizontal line 2

**vline1: float, optional**  
Dashed horizontal line 1

**vline2: float, optional**  
Dashed vertical line 2

**hline: float or Iterable**  
y-axis location of horizontal line(s)

**vline: float or Iterable**  
x-axis location of vertical line(s)

## TDSData.plot\_data

```
TDSData.plot_data(xdata, ydata, *, xheader=None, yheader=None, xlabel=None,  
    ylabel=None, linestyles=None, left=None, right=None, ymin=None,  
    ymax=None, legend=None, grid=False, fig=None, ax=None, latex=True,  
    dpi=None, line_width=1.0, font_size=12, greyscale=False, savefig=None,  
    save_format=None, show=True, title=None, hline1=None, hline2=None,  
    vline1=None, hline=None, vline=None, vline2=None, set_xlim=True,  
    set_ylim=True, autoscale=False, figsize=None, legend_bbox=None,  
    legend_loc=None, legend_ncol=1, mask=True, color=None, style='default',  
    **kwargs)
```

Plot lines for the supplied data and options.

This function takes *xdata* and *ydata* values. If you provide variable indices instead of values, use *plot()*.

See the argument lists of *plot()* for more.

### Parameters

#### xdata

[array-like] An array-like object containing the values for the x-axis variable

#### ydata

[array] An array containing the values of each variables for the y-axis variable.  
The row of *ydata* must match the row of *xdata*. Each column corresponds to a  
variable.

#### mask

[bool] If enabled (1), when specifying axis limits, only data in the limits will be  
used for plotting to optimize for autoscaling. It is done through an index mask.

### Returns

#### (fig, ax)

The figure and axis handles

## Examples

To plot the results of arithmetic calculation of variables, retrieve the values, do the calculation, and plot with *plot\_data*.

```
>>> v = ss.dae.ts.y[:, ss.PVD1.v.a]
>>> Ipcmd = ss.dae.ts.y[:, ss.PVD1.Ipcmd_y.a]
>>> t = ss.dae.ts.t
```

```
>>> ss.TDS.plt.plot_data(t, v * Ipcmd,
>>>                      xlabel='Time [s]',
>>>                      ylabel='Ipcmd [pu]')
```

## TDSDData.plotn

```
TDSDData.plotn(nrows: int, ncols: int, yidxes, xidxes=None, *, dpi=None, titles=None, a=None,
                figsize=None, xlabel=None, ylabel=None, sharex=None, sharey=None,
                show=True, xlabel_offs=(0.5, 0.01), ylabel_offs=(0.05, 0.5), hspace=0.2,
                wspace=0.2, **kwargs)
```

Plot multiple subfigures in one figure.

Parameters *xidxes*, *a*, *xlabels* and *ylabels*, if provided, must have the same length as *yidxes*.

### Parameters

#### nrows

[int] number of rows

#### ncols

[int] number of cols

#### yidx

A list of *BaseVar* or index lists.

## 6.4 I/O

## 6.4.1 andes.io

ANDES input parsers and output formatters.

### Functions

<code>dump(system, output_format[, full_path, ...])</code>	Dump the System data into the requested output format.
<code>get_output_ext(out_format)</code>	Helper function to get the output extension for the given output format.
<code>guess(system)</code>	Guess the input format based on extension and content.
<code>parse(system)</code>	Parse input file with the given format in <code>system.files.input_format</code> .
<code>read_file_like(infile)</code>	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**

<code>andes.io.json</code>	JSON reader and writer for ANDES.
<code>andes.io.matpower</code>	Simple MATPOWER format parser
<code>andes.io.psse</code>	PSS/E file parser.
<code>andes.io.streaming</code>	Simulation data streaming interface for CURENT DiME2.
<code>andes.io.txt</code>	Output TXT file formatter.
<code>andes.io.xlsx</code>	Excel reader and writer for ANDES power system parameters

## andes.io.json

JSON reader and writer for ANDES.

### Functions

<code>read(system, infile)</code>	Read JSON file with ANDES model data into an empty system.
<code>testlines(infile)</code>	
<code>write(system, outfile[, skip_empty, overwrite])</code>	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**

<code>m2mpc(infile)</code>	Parse MATPOWER file and return a dictionary with the data.
<code>mpc2system(mpc, system)</code>	Load an mpc dict into an empty ANDES system.
<code>read(system, file)</code>	Read a MATPOWER data file into mpc, and build andes device elements.
<code>system2mpc(system)</code>	Convert data from an ANDES system to an mpc dict.
<code>testlines(infile)</code>	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 DMR parser.

### Functions

<code>get_block_lines(b, mdata)</code>	Return the number of lines based on the block index in the RAW file.
<code>read(system, file)</code>	Read PSS/E RAW file v32/v33 formats.
<code>read_add(system, file)</code>	Read an addition PSS/E dyr file.
<code>sort_psse_models(dyr_yaml, system)</code>	Sort supported models so that model names are ordered by dependency.
<code>testlines(infile)</code>	Check the raw file for frequency base.

#### `get_block_lines`

`andes.io.psse.get_block_lines (b, mdata)`

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 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

<code>build_init()</code>	Build <i>Varheader</i> , <i>Idxvgs</i> and <i>SysParam</i> after power flow routine
<code>connect()</code>	Connect to DiME 2 server.
<code>finalize()</code>	Send DONE signal when simulation completes
<code>handle_alter(Alter)</code>	Handle parameter altering
<code>handle_event(Event)</code>	Handle Fault, Breaker, Syn and Load Events
<code>record_module_init(name, init_var)</code>	Record the variable requests from modules
<code>send_init([recipient])</code>	Broadcast <i>Varheader</i> , <i>Idxvgs</i> and <i>SysParam</i> to all DiME clients after power flow routine
<code>sync_and_handle()</code>	Sync until the queue is empty.
<code>transpose_matlab_row(a)</code>	
<code>vars_to_modules()</code>	Stream the results from the last step to modules
<code>vars_to_pmu()</code>	Broadcast all PMU measurements and BusFreq measurements in the variable <i>pmudata</i>

### Streaming.build\_init

`Streaming.build_init()`

Build *Varheader*, *Idxvgs* and *SysParam* after power flow routine

### Streaming.connect

`Streaming.connect()`

Connect to DiME 2 server.

If `dime_address` is specified from the command-line, streaming will be automatically enabled. Otherwise, settings from the Config file will be used.

### Streaming.finalize

`Streaming.finalize()`

Send DONE signal when simulation completes

#### Returns

None

### **Streaming.handle.Alter**

`Streaming.handle.Alter` (*Alter*)

Handle parameter altering

### **Streaming.handle.Event**

`Streaming.handle.Event` (*Event*)

Handle Fault, Breaker, Syn and Load Events

### **Streaming.record\_module\_init**

`Streaming.record_module_init` (*name, init\_var*)

Record the variable requests from modules

### **Streaming.send\_init**

`Streaming.send_init` (*recipient='all'*)

Broadcast *Varheader*, *Idxvgs* and *SysParam* to all DiME clients after power flow routine

### **Streaming.sync\_and\_handle**

`Streaming.sync_and_handle()`

Sync until the queue is empty. Handle sync'ed commands.

### **Streaming.transpose\_matlab\_row**

**static** `Streaming.transpose_matlab_row` (*a*)

### **Streaming.vars\_to\_modules**

`Streaming.vars_to_modules()`

Stream the results from the last step to modules

#### **Returns**

None

## Streaming.vars\_to\_pmu

`Streaming.var_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.DataFrame.

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

## 6.5 Interoperability

`andes.interop`

Interoperability package between Andes and other software.

### 6.5.1 andes.interop

Interoperability 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 (2.7.0) 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

<code>from_matpower(m, varname[, system])</code>	Retrieve a MATPOWER mpc case from a MATLAB/Octave instance.
<code>require_matpower(f)</code>	Decorator for functions that require matpower.
<code>to_matpower(m, varname, system)</code>	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 (2.7.0) interface

### Functions

<code>add_gencost(ssp, gen_cost)</code>	Add cost function to converted pandapower net <i>ssp</i> .
<code>build_group_table(ssa, group, columns[, ...])</code>	Build the table for devices in a group in an ANDES System.
<code>make_GSF(ppn[, verify, using_sparse_solver])</code>	Build the Generation Shift Factor matrix of a pandapower net.
<code>make_link_table(ssa)</code>	Build the link table for generators and generator controllers in an ANDES System, including SynGen and DG for now.
<code>require_pandapower(f)</code>	Decorator for functions that require pandapower.
<code>runopp_map(ssp, link_table, **kwargs)</code>	Run OPF in pandapower using pp.runopp() and map results back to ANDES based on the link table.
<code>to_pandapower(ssa[, ctrl, verify, tol])</code>	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=[])`

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*=[], *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

<code>require_pypowsybl(f)</code>	Decorator for functions that require pypowsybl.
<code>to_pypowsybl(ss)</code>	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)
```

(continues on next page)

(continued from previous page)

```
n.get_network_area_diagram()  # show diagram for system  
n.get_single_line_diagram("VL6")  # show single-line diagram for bus 6
```

## 6.6 Others

<code>andes.cli</code>	ANDES command-line interface and argument parsers.
<code>andes.main</code>	Main entry point for the ANDES CLI and scripting interfaces.
<code>andes.utils.paths</code>	Utility functions for loading andes stock test cases
<code>andes.utils.snapshot</code>	Utility functions for saving and loading snapshots.
<code>andes.utils.widgets</code>	Support for Jupyter widgets.

### 6.6.1 andes.cli

ANDES command-line interface and argument parsers.

#### Functions

<code>create_parser()</code>	Create a parser for the command-line interface.
<code>main()</code>	Entry point of the ANDES command-line interface.
<code>preamble()</code>	Log the ANDES command-line preamble at the <code>logging.INFO</code> 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

### 6.6.2 andes.main

Main entry point for the ANDES CLI and scripting interfaces.

## Functions

<code>config_logger([stream_level, stream, file, ...])</code>	Configure an ANDES logger with a <i>FileHandler</i> and a <i>StreamHandler</i> .
<code>demo(**kwargs)</code>	TODO: show some demonstrations from CLI.
<code>doc([attribute, list_supported, config])</code>	Quick documentation from command-line.
<code>edit_conf([edit_config])</code>	Edit the Andes config file which occurs first in the search path.
<code>find_log_path(lg)</code>	Find the file paths of the FileHandlers.
<code>load(case[, codegen, setup, use_input_path])</code>	Load a case and set up a system without running routine.
<code>misc([edit_config, save_config, ...])</code>	Miscellaneous commands.
<code>plot(*args, **kwargs)</code>	Wrapper for the plot tool.
<code>prepare([quick, incremental, models, ...])</code>	Run code generation.
<code>print_license()</code>	Print out Andes license to stdout.
<code>remove_output([recursive])</code>	Remove the outputs generated by Andes, including power flow reports <code>_out.txt</code> , time-domain list <code>_out.lst</code> and data <code>_out.dat</code> , eigenvalue analysis report <code>_eig.txt</code> .
<code>run(filename[, input_path, verbose, ...])</code>	Entry point to run ANDES routines.
<code>run_case(case, *[, routine, profile, ...])</code>	Run single simulation case for the given full path.
<code>save_conf([config_path, overwrite])</code>	Save the Andes config to a file at the path specified by <code>save_config</code> .
<code>selftest([quick, extra])</code>	Run unit tests.
<code>set_logger_level(lg, type_to_set, level)</code>	Set logging level for the given type of handler.
<code>versioninfo()</code>	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, optional] True if logging to `log_file`.

#### log\_file

[str, optional] Log 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` if the function body executes with success. `False` otherwise.

## run

```
andes.main.run(filename, input_path='', verbose=20, mp_verbose=30, ncpu=1, 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 empty string, the save action will not run. Save to `~/.andes/andes.conf` if None.

**Returns****bool**

`True` if 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.

### 6.6.3 andes.utils.paths

Utility functions for loading andes stock test cases

#### Functions

<code>andes_root()</code>	Return the path to the folder of the ANDES package.
<code>cases_root()</code>	Return the root path to the stock cases
<code>confirm_overwrite(outfile[, overwrite])</code>	Confirm overwriting a file.
<code>get_case(rpath[, check])</code>	Return the path to a stock case for a given path relative to <code>andes/cases</code> .
<code>get_config_path([file_name])</code>	Return the path of the config file to be loaded.
<code>get_dot_andes_path()</code>	Return the path to <code>\$HOME/.andes</code>
<code>get_log_dir()</code>	Get the directory for log file.
<code>get_pycode_path([pycode_path, mkdir])</code>	Get the path to the <code>pycode</code> folder.
<code>list_cases([rpath, no_print])</code>	List stock cases under a given folder relative to <code>andes/cases</code>
<code>tests_root()</code>	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])`

---

## DisplayablePath.displayable

```
DisplayablePath.displayable()
```

## DisplayablePath.make\_tree

```
classmethod DisplayablePath.make_tree(root, parent=None, is_last=False,  
criteria=None)
```

## Attributes

---

```
display_filename_prefix_last
```

```
display_filename_prefix_middle
```

```
display_parent_prefix_last
```

```
display_parent_prefix_middle
```

```
displayname
```

---

## DisplayablePath.display\_filename\_prefix\_last

```
DisplayablePath.display_filename_prefix_last = '└—'
```

## DisplayablePath.display\_filename\_prefix\_middle

```
DisplayablePath.display_filename_prefix_middle = '├—'
```

## DisplayablePath.display\_parent\_prefix\_last

```
DisplayablePath.display_parent_prefix_last = '| '
```

### DisplayablePath.display\_parent\_prefix\_middle

```
DisplayablePath.display_parent_prefix_middle = ''
```

### DisplayablePath.displayname

```
property DisplayablePath.displayname
```

## 6.6.4 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.

## **6.6.5 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

<code>edit_sheet(system, model)</code>	Use ipysheet to edit parameters of one model.
<code>edit_system(system)</code>	Edit a loaded ANDES System with ipywidgets.
<code>on_close(b)</code>	Callback for the Close button.
<code>on_update(b)</code>	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 button. Closes ipywidget objects.

### `on_update`

`andes.utils.widgets.on_update(b)`

Callback for the Update button. Sets new parameters back to System.

## BIBLIOGRAPHY

- [Cui2021] H. Cui, F. Li and K. Tomsovic, "Hybrid Symbolic-Numeric Framework for Power System Modeling and Analysis," in IEEE Transactions on Power Systems, vol. 36, no. 2, pp. 1373-1384, March 2021, doi: 10.1109/TPWRS.2020.3017019.
- [Milano2010] F. Milano, "Power System Modelling and Scripting," in Power Modelling and Scripting, F. Milano, Ed. Berlin, Heidelberg: Springer, pp. 202-204, 2010.
- [Sauer] P. W. Sauer, M. A. Pai, and J. H. Chow, Power system dynamics and stability: with synchrophasor measurement and power system toolbox, Second edition. Hoboken, NJ, USA: IEEE Press, Wiley, 2017.
- [PJM5] F. Li and R. Bo, "Small test systems for power system economic studies," IEEE PES General Meeting, 2010, pp. 1-4, doi: 10.1109/PES.2010.5589973.
- [GB] The University of Edinburgh, "Power Systems Test Case Archive", <https://www.maths.ed.ac.uk/optenergy/NetworkData/fullGB>
- [EI] D. Osipov and M. Arrieta-Paternina, "Reduced Eastern Interconnection System Model", [Online]. Available: [https://current.utk.edu/2016SiteVisit/EI\\_LTB\\_Report.pdf](https://current.utk.edu/2016SiteVisit/EI_LTB_Report.pdf).
- [IEEE] University of Washington, "Power Systems Test Case Archive", [Online]. Available: <https://labs.ece.uw.edu/pstca/>
- [RLGC] Qiuhua Huang, "RLGC repository", [Online]. Available: <https://github.com/RLGC-Project/RLGC>
- [MATPOWER] R. D. Zimmerman, "MATPOWER", [Online]. Available: <https://matpower.org/>
- [Nordic] ALSETLab, "Nordpool test system", [Online]. Available: <https://github.com/ALSETLab/Nordic44-Nordpool/tree/master/nordic44/models>
- [PST] Power System Toolbox, [Online]. Available: <https://sites.ecse.rpi.edu/~chowj/PSTMan.pdf>
- [WECC] K. Sun, "Test Cases Library of Power System Sustained Oscillations". Available: <http://web.eecs.utk.edu/~kaisun/Oscillation/basecase.html>
- [PSAT] F. Milano, "Power System Analysis Toolbox", [Online]. Available: <http://faraday1.ucd.ie/psat.html>



## PYTHON MODULE INDEX

### a

andes.cli, 882  
andes.interop, 873  
andes.interop.gridcal, 873  
andes.interop.matpower, 874  
andes.interop.pandapower, 877  
andes.interop.pypowsybl, 880  
andes.io, 862  
andes.io.json, 864  
andes.io.matpower, 865  
andes.io.psse, 867  
andes.io.streaming, 868  
andes.io.txt, 871  
andes.io.xlsx, 871  
andes.main, 883  
andes.plot, 850  
andes.routines, 829  
andes.routines.base, 829  
andes.routines.criteria, 831  
andes.routines.daeint, 831  
andes.routines.eig, 835  
andes.routines.pflow, 842  
andes.routines.tds, 844  
andes.system, 797  
andes.utils.paths, 890  
andes.utils.snapshot, 894  
andes.utils.widgets, 895  
andes.variables, 814  
andes.variables.dae, 815  
andes.variables.fileman, 826  
andes.variables.report, 827



# INDEX

## Symbols

- `__init__()` (*andes.core.discrete.Discrete method*), 368
- `__init__()` (*andes.core.model.Model method*), 279
- `__init__()` (*andes.core.model.ModelCache method*), 290
- `__init__()` (*andes.core.model.ModelCall method*), 291
- `__init__()` (*andes.core.model.ModelData method*), 275
- `__init__()` (*andes.core.param.BaseParam method*), 302
- `__init__()` (*andes.core.param.DataParam method*), 305
- `__init__()` (*andes.core.param.ExtParam method*), 317
- `__init__()` (*andes.core.param.IdxParam method*), 308
- `__init__()` (*andes.core.param.NumParam method*), 313
- `__init__()` (*andes.core.param.TimerParam method*), 320
- `__init__()` (*andes.core.service BaseService method*), 348
- `__init__()` (*andes.core.service.OperationService method*), 350
- `__init__()` (*andes.core.var.Algeb method*), 333
- `__init__()` (*andes.core.var.AliasAlgeb method*), 343
- `__init__()` (*andes.core.var.AliasState method*), 340
- `__init__()` (*andes.core.var.BaseVar method*), 326
- `__init__()` (*andes.core.var.ExtAlgeb method*), 338
- `__init__()` (*andes.core.var.ExtState method*), 335
- `__init__()` (*andes.core.var.ExtVar method*), 328
- `__init__()` (*andes.core.var.State method*), 331
- `__init__()` (*andes.io.streaming.Streaming method*), 868
- `__init__()` (*andes.models.group.GroupBase method*), 268
- `__init__()` (*andes.plot.TDSData method*), 853
- `__init__()` (*andes.routines.base.BaseRoutine method*), 829
- `__init__()` (*andes.routines.daeint.BackEuler method*), 832
- `__init__()` (*andes.routines.daeint.ImplicitIter method*), 833
- `__init__()` (*andes.routines.daeint.Trapezoid method*), 834
- `__init__()` (*andes.routines.eig.EIG method*), 835
- `__init__()` (*andes.routines.pflow.PFlow method*), 842
- `__init__()` (*andes.routines.tds.TDS method*), 845
- `__init__()` (*andes.system.ExistingModels method*), 799
- `__init__()` (*andes.system.System method*), 800
- `__init__()` (*andes.utils.paths.DisplayablePath method*), 892
- `__init__()` (*andes.variables.dae.DAE method*), 816
- `__init__()` (*andes.variables.dae.DAETimeSeries method*), 824
- `__init__()` (*andes.variables.fileman.FileMan method*), 826
- `__init__()` (*andes.variables.report.Report method*), 828

## A

- `a_reset()` (*andes.core.model.Model method*), 281
- `add()` (*andes.core.model.ModelData method*), 275
- `add()` (*andes.core.param.BaseParam method*), 303

add() (*andes.core.param.DataParam method*), 306  
add() (*andes.core.param.ExtParam method*), 317  
add() (*andes.core.param.IdxParam method*), 309  
add() (*andes.core.param.NumParam method*), 314  
add() (*andes.core.param.TimerParam method*), 321  
add() (*andes.models.group.GroupBase method*), 269  
add() (*andes.system.System method*), 802  
add\_callback() (*andes.core.model.ModelCache method*), 291  
add\_gencost() (*in module andes.interop.pandapower*), 877  
add\_model() (*andes.models.group.GroupBase method*), 269  
add\_suffix() (*in module andes.variables.fileman*), 826  
Algeb (*class in andes.core.var*), 332  
AliasAlgeb (*class in andes.core.var*), 343  
AliasState (*class in andes.core.var*), 340  
alloc\_or\_extend\_names() (*in module andes.variables.dae.DAE method*), 817  
alter() (*andes.core.model.Model method*), 281  
andes.cli  
    module, 882  
andes.interop  
    module, 873  
andes.interop.gridcal  
    module, 873  
andes.interop.matpower  
    module, 874  
andes.interop.pandapower  
    module, 877  
andes.interop.pypowsybl  
    module, 880  
andes.io  
    module, 862  
andes.io.json  
    module, 864  
andes.io.matpower  
    module, 865  
andes.io.psse  
    module, 867  
andes.io.streaming  
    module, 868  
andes.io.txt  
    module, 871  
andes.io.xlsx  
    module, 871  
andes.main  
    module, 883  
andes.plot  
    module, 850  
andes.routines  
    module, 829  
andes.routines.base  
    module, 829  
andes.routines.criteria  
    module, 831  
andes.routines.daeint  
    module, 831  
andes.routines.eig  
    module, 835  
andes.routines.pflow  
    module, 842  
andes.routines.tds  
    module, 844  
andes.system  
    module, 797  
andes.utils.paths  
    module, 890  
andes.utils.snapshot  
    module, 894  
andes.utils.widgets  
    module, 895  
andes.variables  
    module, 814  
andes.variables.dae  
    module, 815  
andes.variables.fileman  
    module, 826  
andes.variables.report  
    module, 827  
andes\_root() (*in module andes.utils.paths*), 890  
append\_ijv() (*andes.core.model.ModelCall method*), 292  
as\_df() (*andes.core.model.ModelData method*), 276  
as\_df\_local() (*andes.core.model.ModelData method*), 276  
as\_dict() (*andes.core.model.ModelData method*), 276  
as\_dict() (*andes.system.System method*), 802  
assign\_memory() (*in module andes.core.service BaseService method*), 348

assign_memory()	( <i>andes.core.service.OperationService method</i> ),	check_var()	( <i>andes.core.discrete.Discrete method</i> ),	369
350		class_name	( <i>andes.core.discrete.Discrete property</i> ),	370
<b>B</b>		class_name	( <i>andes.core.model.Model property</i> ),	290
BackEuler ( <i>class in andes.routines.daeint</i> ),	832	class_name	( <i>andes.core.param.BaseParam property</i> ),	305
BaseParam ( <i>class in andes.core.param</i> ),	301	class_name	( <i>andes.core.param.DataParam property</i> ),	308
BaseRoutine ( <i>class in andes.routines.base</i> ),	829	class_name	( <i>andes.core.param.ExtParam property</i> ),	319
BaseService ( <i>class in andes.core.service</i> ),	348	class_name	( <i>andes.core.param.IdxParam property</i> ),	311
BaseVar ( <i>class in andes.core.var</i> ),	325	class_name	( <i>andes.core.param.NumParam property</i> ),	316
bqplot_data() ( <i>andes.plot.TDSData method</i> ),	854	class_name	( <i>andes.core.param.TimerParam property</i> ),	324
build_group_table() ( <i>in module andes.interop.pandapower</i> ),	877	class_name	( <i>andes.core.service BaseService property</i> ),	349
build_init() ( <i>andes.io.streaming.Streaming method</i> ),	869	class_name	( <i>andes.core.service.OperationService property</i> ),	351
build_pattern() ( <i>andes.variables.dae.DAE method</i> ),	817	class_name	( <i>andes.core.var.Algeb property</i> ),	334
<b>C</b>		class_name	( <i>andes.core.var.AliasAlgeb property</i> ),	345
calc_As() ( <i>andes.routines.eig.EIG method</i> ),	836	class_name	( <i>andes.core.var.AliasState property</i> ),	342
calc_eig() ( <i>andes.routines.eig.EIG method</i> ),	836	class_name	( <i>andes.core.var.BaseVar property</i> ),	327
calc_h() ( <i>andes.routines.tds.TDS method</i> ),	846	class_name	( <i>andes.core.var.ExtAlgeb property</i> ),	340
calc_jac() ( <i>andes.routines.daeint.BackEuler static method</i> ),	832	class_name	( <i>andes.core.var.ExtState property</i> ),	337
calc_jac() ( <i>andes.routines.daeint.ImplicitIter static method</i> ),	833	class_name	( <i>andes.core.var.ExtVar property</i> ),	330
calc_jac() ( <i>andes.routines.daeint.Trapezoid static method</i> ),	834	class_name	( <i>andes.core.var.State property</i> ),	332
calc_pfactor() ( <i>andes.routines.eig.EIG method</i> ),	836	class_name	( <i>andes.models.group.GroupBase property</i> ),	273
calc_pu_coeff() ( <i>andes.system.System method</i> ),	803	class_name	( <i>andes.routines.base.BaseRoutine property</i> ),	831
calc_q() ( <i>andes.routines.daeint.BackEuler static method</i> ),	832	class_name	( <i>andes.routines.eig.EIG property</i> ),	841
calc_q() ( <i>andes.routines.daeint.ImplicitIter static method</i> ),	833	class_name	( <i>andes.routines.pflow.PFlow property</i> ),	844
calc_q() ( <i>andes.routines.daeint.Trapezoid static method</i> ),	834	class_name	( <i>andes.routines.tds.TDS property</i> ),	850
call_models() ( <i>andes.system.System method</i> ),	803	clear_arrays()	( <i>andes.variables.dae.DAE method</i> ),	817
cases_root() ( <i>in module andes.utils.paths</i> ),	890	clear_fg()	( <i>andes.variables.dae.DAE method</i> ),	817
check_criteria() ( <i>andes.routines.tds.TDS method</i> ),	846	clear_ijv()	( <i>andes.core.model.ModelCall</i>	
check_eq() ( <i>andes.core.discrete.Discrete method</i> ),	368			
check_iter_err() ( <i>andes.core.discrete.Discrete method</i> ),	368			

*method), 292*

`clear_ijv()` (*andes.variables.dae.DAE method*), [817](#)

`clear_ts()` (*andes.variables.dae.DAE method*), [817](#)

`clear_xy()` (*andes.variables.dae.DAE method*), [818](#)

`clear_z()` (*andes.variables.dae.DAE method*), [818](#)

`collect_config()` (*andes.system.System method*), [803](#)

`collect_ref()` (*andes.system.System method*), [804](#)

`config_logger()` (*in module andes.main*), [884](#)

`confirm_overwrite()` (*in module andes.utils.paths*), [890](#)

`connect()` (*andes.io.streaming.Streaming method*), [869](#)

`connectivity()` (*andes.system.System method*), [804](#)

`create_parser()` (*in module andes.cli*), [882](#)

## D

`DAE` (*class in andes.variables.dae*), [815](#)

`DAETimeSeries` (*class in andes.variables.dae*), [824](#)

`data_to_df()` (*andes.plot.TDSData method*), [854](#)

`DataParam` (*class in andes.core.param*), [305](#)

`deltadelta()` (*in module andes.routines.criteria*), [831](#)

`demo()` (*in module andes.main*), [884](#)

`df` (*andes.variables.dae.DAETimeSeries property*), [826](#)

`Discrete` (*class in andes.core.discrete*), [368](#)

`display_filename_prefix_last` (*andes.utils.paths.DisplayablePath attribute*), [893](#)

`display_filename_prefix_middle` (*andes.utils.paths.DisplayablePath attribute*), [893](#)

`display_parent_prefix_last` (*andes.utils.paths.DisplayablePath attribute*), [893](#)

`display_parent_prefix_middle` (*andes.utils.paths.DisplayablePath attribute*), [894](#)

`displayable` (*andes.utils.paths.DisplayablePath method*), [893](#)

*DisplayablePath* (*class in andes.utils.paths*), [893](#)

`displayname` (*andes.utils.paths.DisplayablePath property*), [894](#)

`do_switch()` (*andes.routines.tds.TDS method*), [846](#)

`doc()` (*andes.core.model.Model method*), [281](#)

`doc()` (*andes.models.group.GroupBase method*), [270](#)

`doc()` (*andes.routines.base.BaseRoutine method*), [830](#)

`doc()` (*andes.routines.eig.EIG method*), [837](#)

`doc()` (*andes.routines.pflow.PFlow method*), [842](#)

`doc()` (*andes.routines.tds.TDS method*), [846](#)

`doc()` (*in module andes.main*), [884](#)

`doc_all()` (*andes.models.group.GroupBase method*), [270](#)

`dump()` (*in module andes.io*), [862](#)

`dump_data()` (*in module andes.io.txt*), [871](#)

## E

`e_clear()` (*andes.core.model.Model method*), [281](#)

`e_clear()` (*andes.system.System method*), [804](#)

`e_code` (*andes.core.var.Algeb attribute*), [334](#)

`e_code` (*andes.core.var.AliasAlgeb attribute*), [345](#)

`e_code` (*andes.core.var.AliasState attribute*), [342](#)

`e_code` (*andes.core.var.ExtAlgeb attribute*), [340](#)

`e_code` (*andes.core.var.ExtState attribute*), [337](#)

`e_code` (*andes.core.var.State attribute*), [332](#)

`edit_conf()` (*in module andes.main*), [885](#)

`edit_sheet()` (*in module andes.utils.widgets*), [896](#)

`edit_system()` (*in module andes.utils.widgets*), [896](#)

`EIG` (*class in andes.routines.eig*), [835](#)

`eig_plot()` (*in module andes.plot*), [850](#)

`example()` (*in module andes.system*), [797](#)

`ExistingModels` (*class in andes.system*), [799](#)

`export_csv()` (*andes.plot.TDSData method*), [854](#)

`export_mat()` (*andes.routines.eig.EIG method*), [837](#)

`ExtAlgeb` (*class in andes.core.var*), [338](#)

`externalize()` (*andes.core.model.Model method*), [282](#)

`ExtParam` (*class in andes.core.param*), [316](#)

`ExtState` (*class in andes.core.var*), [335](#)

`ExtVar` (*class in andes.core.var*), [327](#)

**F**

f\_numeric() (*andes.core.model.Model method*), 282  
f\_update() (*andes.core.model.Model method*), 282  
f\_update() (*andes.system.System method*), 804  
fg (*andes.variables.dae.DAE property*), 822  
fg\_to\_dae() (*andes.system.System method*), 804  
fg\_update() (*andes.routines.pflow.PFlow method*), 843  
fg\_update() (*andes.routines.tds.TDS method*), 847  
FileMan (*class in andes.variables.fileman*), 826  
finalize() (*andes.io.streaming.Streaming method*), 869  
find() (*andes.plot.TDSData method*), 855  
find\_devices() (*andes.system.System method*), 805  
find\_idx() (*andes.core.model.ModelData method*), 276  
find\_idx() (*andes.models.group.GroupBase method*), 270  
find\_log\_path() (*in module andes.main*), 885  
find\_models() (*andes.system.System method*), 805  
find\_param() (*andes.core.model.ModelData method*), 277  
find\_zero\_states() (*andes.routines.eig.EIG method*), 837  
fix\_view\_arrays() (*in module andes.system*), 798  
from\_ipysheet() (*andes.system.System method*), 805  
from\_matpower() (*in module andes.interop.matpower*), 875

**G**

g\_islands() (*andes.system.System method*), 805  
g\_numeric() (*andes.core.model.Model method*), 282  
g\_update() (*andes.core.model.Model method*), 282  
g\_update() (*andes.system.System method*), 806  
get() (*andes.core.model.Model method*), 282  
get() (*andes.models.group.GroupBase method*), 270  
get\_block\_lines() (*in module andes.io.psse*), 867  
get\_call() (*andes.plot.TDSData method*), 855  
get\_case() (*in module andes.utils.paths*), 891  
get\_config\_path() (*in module andes.utils.paths*), 891  
get\_data() (*andes.variables.dae.DAETimeSeries method*), 824  
get\_dot\_andes\_path() (*in module andes.utils.paths*), 891  
get\_field() (*andes.models.group.GroupBase method*), 271  
getFullPath() (*andes.variables.fileman.FileMan method*), 827  
get\_header() (*andes.plot.TDSData method*), 855  
get\_init\_order() (*andes.core.model.Model method*), 283  
get\_inputs() (*andes.core.model.Model method*), 283  
get\_log\_dir() (*in module andes.utils.paths*), 891  
get\_md5() (*andes.core.model.Model method*), 284  
get\_name() (*andes.variables.dae.DAE method*), 818  
get\_names() (*andes.core.discrete.Discrete method*), 369  
get\_names() (*andes.core.param.BaseParam method*), 303  
get\_names() (*andes.core.param.DataParam method*), 306  
get\_names() (*andes.core.param.ExtParam method*), 317  
get\_names() (*andes.core.param.IdxParam method*), 309  
get\_names() (*andes.core.param.NumParam method*), 314  
get\_names() (*andes.core.param.TimerParam method*), 321  
get\_names() (*andes.core.service BaseService method*), 349  
get\_names() (*andes.core.service.OperationService method*), 350  
get\_names() (*andes.core.var.Algeb method*), 333  
get\_names() (*andes.core.var.AliasAlgeb method*), 343  
get\_names() (*andes.core.var.AliasState method*), 341  
get\_names() (*andes.core.var.BaseVar method*), 326  
get\_names() (*andes.core.var.ExtAlgeb method*),

339  
**get\_names()** (*andes.core.var.ExtState method*), 336  
**get\_names()** (*andes.core.var.ExtVar method*), 329  
**get\_names()** (*andes.core.var.State method*), 331  
**get\_next\_idx()** (*andes.models.group.GroupBase method*), 271  
**get\_output\_ext()** (*in module andes.io*), 862  
**get\_property()** (*andes.core.param.BaseParam method*), 304  
**get\_property()** (*andes.core.param.DataParam method*), 307  
**get\_property()** (*andes.core.param.ExtParam method*), 318  
**get\_property()** (*andes.core.param.IdxParam method*), 310  
**get\_property()** (*andes.core.param.NumParam method*), 314  
**get\_property()** (*andes.core.param.TimerParam method*), 322  
**get\_pycode\_path()** (*in module andes.utils.paths*), 892  
**get\_size()** (*andes.variables.dae.DAE method*), 818  
**get\_tex\_names()** (*andes.core.discrete.Discrete method*), 369  
**get\_times()** (*andes.core.model.Model method*), 284  
**get\_values()** (*andes.core.discrete.Discrete method*), 370  
**get\_values()** (*andes.plot.TDSData method*), 856  
**get\_z()** (*andes.system.System method*), 806  
**GroupBase** (*class in andes.models.group*), 268  
**guess()** (*in module andes.io*), 863  
**guess\_event\_time()** (*andes.plot.TDSData method*), 856

**H**

**handle\_alter()** (*andes.io.streaming.Streaming method*), 870  
**handle\_event()** (*andes.io.streaming.Streaming method*), 870

**I**

**idx2model()** (*andes.models.group.GroupBase method*), 271

**idx2uid()** (*andes.core.model.Model method*), 284  
**idx2uid()** (*andes.models.group.GroupBase method*), 272  
**IdxParam** (*class in andes.core.param*), 308  
**ImplicitIter** (*class in andes.routines.daeint*), 833  
**import\_groups()** (*andes.system.System method*), 806  
**import\_models()** (*andes.system.System method*), 806  
**import\_pycode()** (*in module andes.system*), 798  
**import\_routines()** (*andes.system.System method*), 807  
**info** (*andes.variables.report.Report property*), 829  
**init()** (*andes.core.model.Model method*), 284  
**init()** (*andes.routines.base.BaseRoutine method*), 830  
**init()** (*andes.routines.eig.EIG method*), 837  
**init()** (*andes.routines.pflow.PFlow method*), 843  
**init()** (*andes.routines.tds.TDS method*), 847  
**init()** (*andes.system.System method*), 807  
**init\_resume()** (*andes.routines.tds.TDS method*), 847  
**internalize()** (*andes.core.model.Model method*), 284  
**is\_time()** (*andes.core.param.TimerParam method*), 322  
**itm\_step()** (*andes.routines.tds.TDS method*), 847

**J**

**j\_islands()** (*andes.system.System method*), 807  
**j\_numeric()** (*andes.core.model.Model method*), 285  
**j\_update()** (*andes.core.model.Model method*), 285  
**j\_update()** (*andes.system.System method*), 807

**L**

**l\_check\_eq()** (*andes.core.model.Model method*), 285  
**l\_update\_eq()** (*andes.system.System method*), 808  
**l\_update\_var()** (*andes.core.model.Model method*), 285  
**l\_update\_var()** (*andes.system.System method*), 808  
**link\_ext\_param()** (*andes.system.System method*), 808

link\_external() (*andes.core.param.ExtParam method*), 318  
 link\_external() (*andes.core.var.AliasAlgebra method*), 344  
 link\_external() (*andes.core.var.AliasState method*), 341  
 link\_external() (*andes.core.var.ExtAlgebra method*), 339  
 link\_external() (*andes.core.var.ExtState method*), 336  
 link\_external() (*andes.core.var.ExtVar method*), 329  
 list2array() (*andes.core.discrete.Discrete method*), 370  
 list2array() (*andes.core.model.Model method*), 286  
 list\_cases() (*in module andes.utils.paths*), 892  
 load() (*in module andes.main*), 885  
 load\_config\_rc() (*in module andes.system*), 798  
 load\_dae() (*andes.plot.TDSData method*), 856  
 load\_lst() (*andes.plot.TDSData method*), 856  
 load\_npy\_or\_csv() (*andes.plot.TDSData method*), 856  
 load\_plotter() (*andes.routines.tds.TDS method*), 848  
 load\_ss() (*in module andes.utils.snapshot*), 895

## M

m2mpc() (*in module andes.io.matpower*), 865  
 main() (*in module andes.cli*), 883  
 make\_GSF() (*in module andes.interop.pandapower*), 878  
 make\_link\_table() (*in module andes.interop.pandapower*), 878  
 make\_tree() (*andes.utils.paths.DisplayablePath class method*), 893  
 misc() (*in module andes.main*), 886  
 mock\_refresh\_inputs() (*andes.core.model.Model method*), 286  
 Model (*class in andes.core.model*), 277  
 ModelCache (*class in andes.core.model*), 290  
 ModelCall (*class in andes.core.model*), 291  
 ModelData (*class in andes.core.model*), 274  
 module  
     andes.cli, 882  
     andes.interop, 873  
     andes.interop.gridcal, 873

andes.interop.matpower, 874  
 andes.interop.pandapower, 877  
 andes.interop.pypowsybl, 880  
 andes.io, 862  
 andes.io.json, 864  
 andes.io.matpower, 865  
 andes.io.psse, 867  
 andes.io.streaming, 868  
 andes.io.txt, 871  
 andes.io.xlsx, 871  
 andes.main, 883  
 andes.plot, 850  
 andes.routines, 829  
 andes.routines.base, 829  
 andes.routines.criteria, 831  
 andes.routines.daeint, 831  
 andes.routines.eig, 835  
 andes.routines.pflow, 842  
 andes.routines.tds, 844  
 andes.system, 797  
 andes.utils.paths, 890  
 andes.utils.snapshot, 894  
 andes.utils.widgets, 895  
 andes.variables, 814  
 andes.variables.dae, 815  
 andes.variables.fileman, 826  
 andes.variables.report, 827  
 mpc2system() (*in module andes.io.matpower*), 866

## N

n (*andes.core.param.BaseParam property*), 305  
 n (*andes.core.param.DataParam property*), 308  
 n (*andes.core.param.ExtParam property*), 320  
 n (*andes.core.param.IdxParam property*), 311  
 n (*andes.core.param.NumParam property*), 316  
 n (*andes.core.param.TimerParam property*), 324  
 n (*andes.core.service BaseService property*), 349  
 n (*andes.core.service.OperationService property*), 351  
 n (*andes.models.group.GroupBase property*), 273  
 newton\_krylov() (*andes.routines.pflow.PFlow method*), 843  
 nr\_solve() (*andes.routines.pflow.PFlow method*), 843  
 nr\_step() (*andes.routines.pflow.PFlow method*), 843  
 numba\_jitify() (*andes.core.model.Model method*), 286

NumParam (*class in andes.core.param*), 311

## O

on\_close () (*in module andes.utils.widgets*), 896

on\_update () (*in module andes.utils.widgets*), 896

OperationService (*class in andes.core.service*), 349

## P

panoview () (*andes.plot.TDSData method*), 857

parse () (*in module andes.io*), 863

parse\_y () (*in module andes.plot*), 851

PFlow (*class in andes.routines.pflow*), 842

plot () (*andes.plot.TDSData method*), 857

plot () (*andes.routines.eig.EIG method*), 837

plot () (*in module andes.main*), 886

plot\_data () (*andes.plot.TDSData method*), 860

plot\_root\_loci () (*andes.routines.eig.EIG method*), 839

plotn () (*andes.plot.TDSData method*), 861

post\_init\_check () (*andes.core.model.Model method*), 286

post\_process () (*andes.routines.eig.EIG method*), 840

preamble () (*in module andes.cli*), 883

precompile () (*andes.core.model.Model method*), 286

precompile () (*andes.system.System method*), 808

prepare () (*andes.core.model.Model method*), 286

prepare () (*andes.system.System method*), 808

prepare () (*in module andes.main*), 886

print\_array () (*andes.variables.dae.DAE method*), 818

print\_license () (*in module andes.main*), 887

## R

r\_code (*andes.core.var.AliasAlgeb attribute*), 345

r\_code (*andes.core.var.AliasState attribute*), 342

r\_code (*andes.core.var.ExtAlgeb attribute*), 340

r\_code (*andes.core.var.ExtState attribute*), 337

read () (*in module andes.io.json*), 864

read () (*in module andes.io.matpower*), 866

read () (*in module andes.io.psse*), 867

read () (*in module andes.io.xlsx*), 872

read\_add () (*in module andes.io.psse*), 867

read\_file\_like () (*in module andes.io*), 863

record\_module\_init () (*andes.io.streaming.Streaming method*), 870

refresh () (*andes.core.model.ModelCache method*), 291

refresh\_inputs () (*andes.core.model.Model method*), 287

refresh\_inputs\_arg () (*andes.core.model.Model method*), 287

register\_debug\_equation () (*andes.core.model.Model method*), 287

reload () (*andes.system.System method*), 809

reload\_submodules () (*in module andes.system*), 799

remove\_output () (*in module andes.main*), 887

remove\_pycapsule () (*andes.system.System method*), 809

Report (*class in andes.variables.report*), 828

report () (*andes.routines.base.BaseRoutine method*), 830

report () (*andes.routines.eig.EIG method*), 840

report () (*andes.routines.pflow.PFlow method*), 844

report () (*andes.routines.tds.TDS method*), 848

report\_info () (*in module andes.variables.report*), 827

request\_address () (*andes.variables.dae.DAE method*), 819

require\_gridcal () (*in module andes.interop.gridcal*), 874

require\_matpower () (*in module andes.interop.matpower*), 876

require\_pandapower () (*in module andes.interop.pandapower*), 879

require\_pypowsybl () (*in module andes.interop.pypowsybl*), 881

reset () (*andes.core.var.Algeb method*), 333

reset () (*andes.core.var.AliasAlgeb method*), 344

reset () (*andes.core.var.AliasState method*), 341

reset () (*andes.core.var.BaseVar method*), 326

reset () (*andes.core.var.ExtAlgeb method*), 339

reset () (*andes.core.var.ExtState method*), 336

reset () (*andes.core.var.ExtVar method*), 329

reset () (*andes.core.var.State method*), 331

reset () (*andes.routines.tds.TDS method*), 848

reset () (*andes.system.System method*), 810

reset () (*andes.variables.dae.DAE method*), 819

reset () (*andes.variables.dae.DAETimeSeries method*), 825

resize\_arrays () (*andes.variables.dae.DAE method*), 819

restore() (*andes.core.param.ExtParam method*), 318  
 restore() (*andes.core.param.NumParam method*), 315  
 restore() (*andes.core.param.TimerParam method*), 322  
 restore\_sparse() (*andes.variables.dae.DAE method*), 820  
 rewind() (*andes.routines.tds.TDS method*), 848  
 run() (*andes.routines.base.BaseRoutine method*), 830  
 run() (*andes.routines.eig.EIG method*), 840  
 run() (*andes.routines.pflow.PFlow method*), 844  
 run() (*andes.routines.tds.TDS method*), 848  
 run() (*in module andes.main*), 887  
 run\_case() (*in module andes.main*), 888  
 runopp\_map() (*in module andes.interop.pandapower*), 879

**S**

s\_numeric() (*andes.core.model.Model method*), 287  
 s\_numeric\_var() (*andes.core.model.Model method*), 287  
 s\_update() (*andes.core.model.Model method*), 287  
 s\_update\_post() (*andes.core.model.Model method*), 288  
 s\_update\_post() (*andes.system.System method*), 810  
 s\_update\_var() (*andes.core.model.Model method*), 288  
 s\_update\_var() (*andes.system.System method*), 810  
 save\_conf() (*in module andes.main*), 889  
 save\_config() (*andes.system.System method*), 810  
 save\_output() (*andes.routines.tds.TDS method*), 848  
 save\_ss() (*in module andes.utils.snapshot*), 895  
 selftest() (*in module andes.main*), 889  
 send\_init() (*andes.io.streaming.Streaming method*), 870  
 set() (*andes.core.model.Model method*), 288  
 set() (*andes.core.param.BaseParam method*), 304  
 set() (*andes.core.param.DataParam method*), 307  
 set() (*andes.core.param.ExtParam method*), 318  
 set() (*andes.core.param.IdxParam method*), 310

set() (*andes.core.param.NumParam method*), 315  
 set() (*andes.core.param.TimerParam method*), 323  
 set() (*andes.models.group.GroupBase method*), 272  
 set() (*andes.variables.fileman.FileMan method*), 827  
 set\_address() (*andes.core.var.Algeb method*), 334  
 set\_address() (*andes.core.var.AliasAlgeb method*), 344  
 set\_address() (*andes.core.var.AliasState method*), 342  
 set\_address() (*andes.core.var.BaseVar method*), 326  
 set\_address() (*andes.core.var.ExtAlgeb method*), 339  
 set\_address() (*andes.core.var.ExtState method*), 336  
 set\_address() (*andes.core.var.ExtVar method*), 329  
 set\_address() (*andes.core.var.State method*), 331  
 set\_address() (*andes.system.System method*), 811  
 set\_all() (*andes.core.param.BaseParam method*), 304  
 set\_all() (*andes.core.param.DataParam method*), 307  
 set\_all() (*andes.core.param.ExtParam method*), 319  
 set\_all() (*andes.core.param.IdxParam method*), 310  
 set\_all() (*andes.core.param.NumParam method*), 315  
 set\_all() (*andes.core.param.TimerParam method*), 323  
 set\_arrays() (*andes.core.var.Algeb method*), 334  
 set\_arrays() (*andes.core.var.AliasAlgeb method*), 344  
 set\_arrays() (*andes.core.var.AliasState method*), 342  
 set\_arrays() (*andes.core.var.BaseVar method*), 327  
 set\_arrays() (*andes.core.var.ExtAlgeb method*), 339  
 set\_arrays() (*andes.core.var.ExtState method*), 336

set\_arrays() (*andes.core.var.ExtVar method*), 329  
 set\_arrays() (*andes.core.var.State method*), 332  
 set\_backref() (*andes.core.model.Model method*), 288  
 set\_backref() (*andes.models.group.GroupBase method*), 273  
 set\_config() (*andes.system.System method*), 811  
 set\_dae\_names() (*andes.system.System method*), 811  
 set\_font() (*in module andes.plot*), 851  
 set\_in\_use() (*andes.core.model.Model method*), 289  
 set\_latex() (*in module andes.plot*), 851  
 set\_logger\_level() (*in module andes.main*), 889  
 set\_method() (*andes.routines.tds.TDS method*), 849  
 set\_output\_subidx() (*andes.system.System method*), 811  
 set\_pu\_coeff() (*andes.core.param.ExtParam method*), 319  
 set\_pu\_coeff() (*andes.core.param.NumParam method*), 315  
 set\_pu\_coeff() (*andes.core.param.TimerParam method*), 323  
 set\_style() (*in module andes.plot*), 852  
 set\_t() (*andes.variables.dae.DAE method*), 820  
 set\_var\_arrays() (*andes.system.System method*), 811  
 setup() (*andes.system.System method*), 812  
 solve\_iter() (*andes.core.model.Model method*), 289  
 solve\_iter\_single() (*andes.core.model.Model method*), 289  
 sort\_psse\_models() (*in module andes.io.psse*), 868  
 State (*class in andes.core.var*), 330  
 stats() (*andes.routines.eig.EIG method*), 840  
 step() (*andes.routines.daeint.BackEuler static method*), 832  
 step() (*andes.routines.daeint.ImplicitIter static method*), 833  
 step() (*andes.routines.daeint.Trapezoid static method*), 834  
 store() (*andes.variables.dae.DAE method*), 820  
 store\_adder\_setter() (*andes.system.System method*), 812  
 store\_existing() (*andes.system.System method*), 812  
 store\_no\_check\_init() (*andes.system.System method*), 812  
 store\_sparse\_ijv() (*andes.variables.dae.DAE method*), 820  
 store\_sparse\_pattern() (*andes.core.model.Model method*), 289  
 store\_sparse\_pattern() (*andes.system.System method*), 812  
 store\_switch\_times() (*andes.system.System method*), 813  
 Streaming (*class in andes.io.streaming*), 868  
 streaming\_init() (*andes.routines.tds.TDS method*), 849  
 streaming\_step() (*andes.routines.tds.TDS method*), 849  
 summary() (*andes.routines.base.BaseRoutine method*), 830  
 summary() (*andes.routines.eig.EIG method*), 840  
 summary() (*andes.routines.pflow.PFlow method*), 844  
 summary() (*andes.routines.tds.TDS method*), 849  
 summary() (*andes.system.System method*), 813  
 supported\_models() (*andes.system.System method*), 813  
 sweep() (*andes.routines.eig.EIG method*), 840  
 switch\_action() (*andes.core.model.Model method*), 290  
 switch\_action() (*andes.system.System method*), 813  
 sync\_and\_handle() (*andes.io.streaming.Streaming method*), 870  
 System (*class in andes.system*), 799  
 system2mpc() (*in module andes.io.matpower*), 866

## T

t\_const (*andes.core.var.AliasState attribute*), 343  
 t\_const (*andes.core.var.ExtState attribute*), 337  
 TDS (*class in andes.routines.tds*), 845  
 TDSDData (*class in andes.plot*), 853  
 tdsplot() (*in module andes.plot*), 852  
 test\_init() (*andes.routines.tds.TDS method*), 849  
 testlines() (*in module andes.io.json*), 864  
 testlines() (*in module andes.io.matpower*), 866  
 testlines() (*in module andes.io.psse*), 868

<code>testlines()</code> ( <i>in module andes.io.xlsx</i> ), 872	<code>v_numeric()</code> ( <i>andes.core.model.Model method</i> ), 290
<code>tests_root()</code> ( <i>in module andes.utils.paths</i> ), 892	<code>vars_to_dae()</code> ( <i>andes.system.System method</i> ), 814
<code>TimerParam</code> ( <i>class in andes.core.param</i> ), 320	<code>vars_to_models()</code> ( <i>andes.system.System method</i> ), 814
<code>to_array()</code> ( <i>andes.core.param.ExtParam method</i> ), 319	<code>vars_to_modules()</code> ( <i>andes.io.streaming.Streaming method</i> ), 870
<code>to_array()</code> ( <i>andes.core.param.NumParam method</i> ), 316	<code>vars_to_pmu()</code> ( <i>andes.io.streaming.Streaming method</i> ), 871
<code>to_array()</code> ( <i>andes.core.param.TimerParam method</i> ), 324	<code>versioninfo()</code> ( <i>in module andes.main</i> ), 889
<code>to_gridcal()</code> ( <i>in module andes.interop.gridcal</i> ), 874	<b>W</b>
<code>to_ipysheet()</code> ( <i>andes.system.System method</i> ), 814	<code>warn_init_limit()</code> ( <i>andes.core.discrete.Discrete method</i> ), 370
<code>to_matpower()</code> ( <i>in module andes.interop.matpower</i> ), 876	<code>write()</code> ( <i>andes.variables.report.Report method</i> ), 828
<code>to_pandapower()</code> ( <i>in module andes.interop.pandapower</i> ), 879	<code>write()</code> ( <i>in module andes.io.json</i> ), 864
<code>to_pypowsybl()</code> ( <i>in module andes.interop.pypowsybl</i> ), 881	<code>write()</code> ( <i>in module andes.io.xlsx</i> ), 872
<code>transpose_matlab_row()</code> ( <i>andes.io.streaming.Streaming static method</i> ), 870	<code>write_1st()</code> ( <i>andes.variables.dae.DAE method</i> ), 821
<code>Trapezoid</code> ( <i>class in andes.routines.daeint</i> ), 834	<code>write_npy()</code> ( <i>andes.variables.dae.DAE method</i> ), 821
<b>U</b>	<code>write_npz()</code> ( <i>andes.variables.dae.DAE method</i> ), 821
<code>undill()</code> ( <i>andes.system.System method</i> ), 814	<b>X</b>
<code>unpack()</code> ( <i>andes.variables.dae.DAETimeSeries method</i> ), 825	<code>x_name_output</code> ( <i>andes.variables.dae.DAE property</i> ), 822
<code>unpack_df()</code> ( <i>andes.variables.dae.DAETimeSeries method</i> ), 825	<code>x_tex_name_output</code> ( <i>andes.variables.dae.DAE property</i> ), 822
<code>unpack_np()</code> ( <i>andes.variables.dae.DAETimeSeries method</i> ), 825	<code>xy</code> ( <i>andes.variables.dae.DAE property</i> ), 823
<code>update()</code> ( <i>andes.variables.report.Report method</i> ), 828	<code>xy_name</code> ( <i>andes.variables.dae.DAE property</i> ), 823
<code>update_from_df()</code> ( <i>andes.core.model.ModelData method</i> ), 277	<code>xy_tex_name</code> ( <i>andes.variables.dae.DAE property</i> ), 823
<b>V</b>	<code>xyz</code> ( <i>andes.variables.dae.DAE property</i> ), 823
<code>v</code> ( <i>andes.core.service.OperationService property</i> ), 351	<code>xyz_name</code> ( <i>andes.variables.dae.DAE property</i> ), 823
<code>v_code</code> ( <i>andes.core.var.Algeb attribute</i> ), 335	<code>xyz_tex_name</code> ( <i>andes.variables.dae.DAE property</i> ), 823
<code>v_code</code> ( <i>andes.core.var.AliasAlgeb attribute</i> ), 345	<b>Y</b>
<code>v_code</code> ( <i>andes.core.var.AliasState attribute</i> ), 343	<code>y_name_output</code> ( <i>andes.variables.dae.DAE property</i> ), 823
<code>v_code</code> ( <i>andes.core.var.ExtAlgeb attribute</i> ), 340	<code>y_tex_name_output</code> ( <i>andes.variables.dae.DAE property</i> ), 824
<code>v_code</code> ( <i>andes.core.var.ExtState attribute</i> ), 337	<b>Z</b>
<code>v_code</code> ( <i>andes.core.var.State attribute</i> ), 332	<code>zip_ijv()</code> ( <i>andes.core.model.ModelCall method</i> ), 292