

**Long-Term Dynamic Simulation of Power Systems
using Python, Agent Based Modeling, and
Time-Sequenced Power Flows**

by
Thad Haines

A thesis submitted in partial fulfillment of the
requirements for the degree of

Master of Science in Electrical Engineering

Montana Technological University
2020



Abstract

Automated controls facilitate reliable and efficient operation of modern power systems. Engineers employ computer simulations to develop, analyze, and tune such controls. Short-term dynamic, or transient, power system simulation is a useful and standardized power industry tool. Researchers have developed effective long-term dynamic (LTD) simulators, but there is not yet an industry standard computational method or software package for LTD simulation.

This work introduces a novel LTD simulation tool and provides examples of various engineering applications. The newly created software tool, Power System Long-Term Dynamic Simulator (PSLTDSim), uses a time-sequenced power flow (TSPF) technique to simulate LTD events. The TSPF technique incorporates a number of modeling assumptions that simplify certain engineering calculations. Despite such simplifications, PSLTDSim demonstrates an acceptable amount of accuracy for ramp and small step type perturbations when compared to industry standard transient simulation software. Demonstrated PSLTDSim engineering applications include: investigation of long-term governor deadband effects, automatic generation control tuning, and switched shunt coordination during multi-hour events. Further demonstrated examples consist of user modified turbine speed governor behavior and variable system damping and inertia. This material is based upon work supported by the U.S. Department of Energy, Office of Science, Basic Energy Sciences, under Award Number DE-SC0012671.

Keywords: Power system simulation, Long-term dynamics, Time-sequenced power flow, Automatic generation control, AGC, Agent based modeling, ABM, Advanced message queueing protocol, AMQP, Python, IronPython

Dedication

Forthcoming...

Acknowledgments

Forthcoming...

Table of Contents

Abstract	ii
Dedication	iii
Acknowledgments	iv
Table of Contents	v
List of Tables	x
List of Figures	xi
List of Equations	xviii
Glossary of Terms	xix
1 Introduction	1
2 Electrical Engineering Background	3
2.1 Power System Basics	3
2.2 Power Flow	8
2.3 The Swing Equation	10
2.4 Turbine Speed Governors	11
2.5 Automatic Generation Control	12
2.6 Reactive Power and Voltage Control	13
3 Software Background	14
3.1 Classical Transient Stability Simulation	14
3.2 Python	14
3.2.1 Python Packages	14
3.2.2 Varieties of Python	15
3.2.3 Python Specific Data Types	15
3.3 Advanced Message Queueing Protocol	16
3.4 Agent Based Modeling	16
4 Software Tool	18
4.1 Time-Sequenced Power Flows	18
4.2 Simulation Assumptions and Simplifications	20

4.2.1	General Assumptions and Simplifications	20
4.2.2	Time Step Assumptions and Simplifications	20
4.2.3	Combined System Frequency	21
4.2.4	Distribution of Accelerating Power	22
4.2.5	Governor models	22
4.2.5.1	Casting Process for genericGov	23
4.3	General Software Explanation	25
4.3.1	Interprocess Communication	26
4.3.2	Simulation Inputs	26
4.3.2.1	PSLF Compatible Input	26
4.3.2.2	Simulation Parameter Input (.py)	27
4.3.2.3	Long-Term Dynamic Input (.ltd.py)	28
4.3.2.3.1	Perturbance List	28
4.3.2.3.2	Noise Agent Attribute	30
4.3.2.3.3	Balancing Authority Dictionary	31
4.3.2.3.4	Load Control Dictionary	31
4.3.2.3.5	Generation Control Dictionary	33
4.3.2.3.6	Governor Input Delay and Filtering Dictionary . .	34
4.3.2.3.7	Governor Deadband Dictionary	35
4.3.2.3.8	Definite Time Controller Dictionary	36
4.3.3	Simulation Initialization	38
4.3.3.1	Process Creation	38
4.3.3.2	Mirror Initialization	38
4.3.3.3	Dynamic Initialization Pre-Simulation Loop	40
4.3.4	Simulation Loop	42
4.3.5	Simulation Outputs	44
4.4	Software Validation	45
4.4.1	Validation Plots Explained	45
4.4.1.1	Comparison Plot	46
4.4.1.2	Difference Plot	46
4.4.1.3	Percent Difference Plot	47
4.4.1.4	Weighted Frequency Plot	48
4.4.2	Six Machine System	50
4.4.2.1	Simulated Scenario Descriptions	50
4.4.2.2	Frequency Results	51
4.4.2.3	Generator Mechanical Power Results	52

4.4.2.4	Generator Real Power Results	53
4.4.2.5	Voltage Magnitude Results	54
4.4.2.6	Voltage Angle Results	56
4.4.2.7	Generator Reactive Power Results	57
4.4.2.8	Branch Current Results	58
4.4.2.9	Branch Real Power Flow Results	60
4.4.2.10	Branch Reactive Power Flow Results	61
4.4.2.11	Six Machine Result Summary	62
4.4.3	Mini WECC System	63
4.4.3.1	Simulated Scenario Descriptions	63
4.4.3.2	Frequency Results	65
4.4.3.3	Generator Mechanical Power Results	66
4.4.3.4	Generator Real Power Results	67
4.4.3.5	Voltage Magnitude Results	68
4.4.3.6	Voltage Angle Results	70
4.4.3.7	Generator Reactive Power Results	71
4.4.3.8	Branch Current Results	72
4.4.3.9	Branch Real Power Flow Results	73
4.4.3.10	Branch Reactive Power Flow Results	75
4.4.3.11	Mini WECC Result Summary	76
4.4.4	Mini WECC with PSS System	77
4.4.4.1	Simulated Scenario Descriptions	77
4.4.4.2	Frequency Results	77
4.4.4.3	Generator Mechanical Power Results	78
4.4.4.4	Generator Real Power Results	80
4.4.4.5	Voltage Magnitude Results	81
4.4.4.6	Voltage Angle Results	82
4.4.4.7	Generator Reactive Power Results	83
4.4.4.8	Branch Current Results	84
4.4.4.9	Branch Real Power Flow Results	86
4.4.4.10	Branch Reactive Power Flow Results	87
4.4.4.11	Mini WECC with PSS Result Summary	88
4.4.5	Full WECC System	89
4.4.5.1	Load Step	90
4.4.6	Validation Summary	92

5 Engineering Applications	93
5.1 Simulated Events of Interest	93
5.2 Relevant NERC Standards	94
5.2.1 BAL-001-2	94
5.2.2 BAL-002-3	95
5.2.3 BAL-003-1.1	96
5.2.4 NERC Standard Summary	96
5.3 Simulated Balancing Authority Controls	97
5.3.1 Governor Deadbands	98
5.3.2 Area Wide Governor Droops	99
5.3.3 Automatic Generation Control	99
5.3.3.1 Frequency Bias	99
5.3.3.2 Integral of Area Control Error	100
5.3.3.3 Conditional Area Control Error Summing	100
5.3.3.4 Area Control Error Filtering	101
5.3.3.5 Controlled Generators and Participation Factors	101
5.4 Governor Deadband Effect on Valve Travel	103
5.4.1 Governor Deadband Simulation Configuration	103
5.4.2 Governor Deadband Simulation Results	104
5.5 Automatic Generation Control Tuning	106
5.5.1 AGC Simulation Configuration	106
5.5.2 AGC Simulation Results	107
5.5.2.1 Base Case Results	107
5.5.2.2 AGC Tuning Results	107
5.5.2.3 Noise and Deadband Simulation Results	109
5.5.2.4 Conditional ACE Results	110
5.6 Long-Term Simulation with Shunt Control	114
5.6.1 Morning Peak Forecast Demand Simulation	114
5.6.1.1 Morning Peak Forecast Demand Results	115
5.6.2 Virtual Wind Ramp Simulation	120
5.6.2.1 Virtual Wind Ramp Results	120
5.6.3 Long-Term Simulation with Shunt Control Result Summary	125
5.7 Feed-Forward Governor Action	126
5.7.1 Feed-Forward Governor Simulation Configuration	126
5.7.2 Feed-Forward Governor Simulation Results	128
5.8 Variable System Damping and Inertia	129

5.8.1	Damping and Inertia Simulation Configuration	129
5.8.2	Damping and Inertia Simulation Results	130
6	Conclusions	133
6.1	Simulation Conclusions	133
6.2	Engineering Application Conclusions	133
7	Future Work	134
8	Bibliography	135
A	Six Machine System Details	140
B	Code Examples	144
C	Large Tables	152
D	Detailed Valve Travel Results	154
E	Additional AGC Results	159

List of Tables

4.1	Generic governor model casting between LTD and PSDS.	23
4.2	Generic governor model parameters.	24
4.3	Perturbation agent identification options.	29
4.4	Perturbation agent action options.	29
4.5	WECC machine model count and capacity.	89
4.6	WECC governor models used in PSDS.	90
4.7	WECC governor models used in PSLTDSim.	90
5.1	Tie-line bias AGC type ACE calculations.	101
5.2	Total valve travel for various deadband scenarios.	105
A.1	Six machine bus table.	140
A.2	Six machine line table.	141
A.3	Six machine transformer table.	141
A.4	Six machine generator table.	141
A.5	Six machine load table.	141
A.6	Six machine shunt table.	142
C.1	Balancing authority dictionary input information.	152
C.2	Simulation parameters dictionary input information.	153

List of Figures

2.1	General form of a modern power system.	3
2.2	Location and size of US electric generation.	4
2.3	US Interconnections.	5
2.4	Frequency relation to electric supply and demand.	6
2.5	Classic frequency and automatic control response.	7
2.6	Two buses with power flow between them.	8
2.7	Ideal governor droop action.	11
4.1	Time-sequenced power flows visualized.	18
4.2	CTS and TSPF data output comparison.	19
4.3	Block diagram of modified tgov1 model.	23
4.4	Block diagram of genericGov model.	23
4.5	High level software flow chart.	25
4.6	An example of a simParams dictionary.	27
4.7	Perturbation agent examples.	30
4.8	Noise agent creation example.	30
4.9	Load control agent dictionary definition example.	32
4.10	Generation control agent dictionary definition example.	34
4.11	Block diagram of delay block.	34
4.12	Governor delay dictionary definition example.	35
4.13	Governor deadband dictionary definition example.	36
4.14	Definite time controller dictionary definition example.	37
4.15	Simulation time step flowchart.	43
4.16	Validation plot examples.	45
4.17	Comparison plot examples.	46
4.18	Difference plot example.	46
4.19	Percent difference plot examples.	47
4.20	Frequency comparison plot example.	49
4.21	Six machine system.	50
4.22	Six machine load step system frequency comparison.	51
4.23	Six machine load ramp system frequency comparison.	51
4.24	Six machine generator trip system frequency comparison.	52
4.25	Six machine load step mechanical power comparison.	52
4.26	Six machine load ramp mechanical power comparison.	53
4.27	Six machine generator trip mechanical power comparison.	53
4.28	Six machine load step real power comparison.	54

4.29	Six machine load ramp real power comparison.	54
4.30	Six machine generator trip real power comparison.	54
4.31	Six machine load step voltage comparison.	55
4.32	Six machine load ramp voltage comparison.	55
4.33	Six machine generator trip voltage comparison.	56
4.34	Six machine load step voltage angle comparison.	56
4.35	Six machine load ramp voltage angle comparison.	57
4.36	Six machine generator trip voltage angle comparison.	57
4.37	Six machine load step reactive power comparison.	58
4.38	Six machine load ramp reactive power comparison.	58
4.39	Six machine generator trip reactive power comparison.	58
4.40	Six machine load step branch current flow comparison.	59
4.41	Six machine load ramp branch current flow comparison.	59
4.42	Six machine generator trip branch current flow comparison.	59
4.43	Six machine load step branch real power flow comparison.	60
4.44	Six machine load ramp branch real power flow comparison.	60
4.45	Six machine generator trip branch real power flow comparison.	61
4.46	Six machine load step branch reactive power flow comparison.	61
4.47	Six machine load ramp branch reactive power flow comparison.	62
4.48	Six machine generator trip branch reactive power flow comparison.	62
4.49	Mini WECC system.	64
4.50	Mini WECC load step system frequency comparison.	65
4.51	Mini WECC load ramp system frequency comparison.	65
4.52	Mini WECC generator trip system frequency comparison.	66
4.53	Mini WECC load step mechanical power comparison.	66
4.54	Mini WECC load ramp mechanical power comparison.	67
4.55	Mini WECC generator trip mechanical power comparison.	67
4.56	Mini WECC load step real power comparison.	68
4.57	Mini WECC load ramp real power comparison.	68
4.58	Mini WECC generator trip real power comparison.	68
4.59	Mini WECC load step voltage comparison.	69
4.60	Mini WECC load ramp voltage comparison.	69
4.61	Mini WECC generator trip voltage comparison.	69
4.62	Mini WECC load step voltage angle comparison.	70
4.63	Mini WECC load ramp voltage angle comparison.	70
4.64	Mini WECC generator trip voltage angle comparison.	71

4.65	Mini WECC load step reactive power comparison.	71
4.66	Mini WECC load ramp reactive power comparison.	72
4.67	Mini WECC generator trip reactive power comparison.	72
4.68	Mini WECC load step branch current flow comparison.	73
4.69	Mini WECC load ramp branch current flow comparison.	73
4.70	Mini WECC generator trip branch current flow comparison.	73
4.71	Mini WECC load step branch real power flow comparison.	74
4.72	Mini WECC load ramp branch real power flow comparison.	74
4.73	Mini WECC generator trip branch real power flow comparison.	74
4.74	Mini WECC load step branch reactive power flow comparison.	75
4.75	Mini WECC load ramp branch reactive power flow comparison.	75
4.76	Mini WECC generator trip branch reactive power flow comparison.	76
4.77	Mini WECC with PSS load step system frequency comparison.	78
4.78	Mini WECC with PSS load ramp system frequency comparison.	78
4.79	Mini WECC with PSS generator trip system frequency comparison.	78
4.80	Mini WECC with PSS load step mechanical power comparison.	79
4.81	Mini WECC with PSS load ramp mechanical power comparison.	79
4.82	Mini WECC with PSS generator trip mechanical power comparison.	79
4.83	Mini WECC with PSS load step real power comparison.	80
4.84	Mini WECC with PSS load ramp real power comparison.	80
4.85	Mini WECC with PSS generator trip real power comparison.	81
4.86	Mini WECC with PSS load step voltage comparison.	81
4.87	Mini WECC with PSS load ramp voltage comparison.	82
4.88	Mini WECC with PSS generator trip voltage comparison.	82
4.89	Mini WECC with PSS load step voltage angle comparison.	83
4.90	Mini WECC with PSS load ramp voltage angle comparison.	83
4.91	Mini WECC with PSS generator trip voltage angle comparison.	83
4.92	Mini WECC with PSS load step reactive power comparison.	84
4.93	Mini WECC with PSS load ramp reactive power comparison.	84
4.94	Mini WECC with PSS generator trip reactive power comparison.	84
4.95	Mini WECC with PSS load step branch current flow comparison.	85
4.96	Mini WECC with PSS load ramp branch current flow comparison.	85
4.97	Mini WECC with PSS generator trip branch current flow comparison.	85
4.98	Mini WECC with PSS load step branch real power flow comparison.	86
4.99	Mini WECC with PSS load ramp branch real power flow comparison.	86
4.100	Mini WECC with PSS generator trip branch real power flow comparison.	87

4.101	Mini WECC with PSS load step branch reactive power flow comparison.	87
4.102	Mini WECC with PSS load ramp branch reactive power flow comparison.	88
4.103	Mini WECC with PSS generator trip branch reactive power flow comparison.	88
4.104	Full WECC frequency comparison.	91
4.105	Full WECC absolute frequency difference.	91
5.1	Balancing authority ACE limit for different values of B.	95
5.2	Single sysBA dictionary definition.	97
5.3	Examples of available deadband action.	98
5.4	Block diagram of ACE calculation and manipulation.	99
5.5	Block diagrams of optional ACE filters.	101
5.6	Cumulative system change in load for governor deadband simulation.	103
5.7	System frequency comparison of different deadband scenarios.	104
5.8	Detail comparison of initial valve movement.	104
5.9	Area valve travel over time for homogeneous and non-homogeneous scenarios.	105
5.10	Random noise added to AGC simulations.	106
5.11	Frequency response to generation loss event in area 1.	107
5.12	Calculated BA values during generation loss event in area 1.	107
5.13	AGC Frequency response to area 1 base case scenario.	108
5.14	Calculated BA values during area 1 AGC tuning.	108
5.15	Area 1 controlled generation response during AGC tuning.	108
5.16	Area 2 controlled generation response during AGC tuning.	109
5.17	AGC frequency response with noise and deadbands.	109
5.18	Calculated BA values with noise and deadbands.	110
5.19	Area 1 controlled generation response to noise and deadbands.	110
5.20	Area 2 controlled generation response to noise and deadbands.	110
5.21	Frequency response to event using TLB 0.	111
5.22	Calculated BA values during an event using TLB 0.	111
5.23	Area 1 controlled generation response to internal area event using TLB 0.	111
5.24	Area 2 controlled generation response to external area event using TLB 0.	112
5.25	Frequency response to event using TLB 4.	112
5.26	Calculated BA values during an event using TLB 4.	112
5.27	Area 1 controlled generation response to internal area event using TLB 4.	113
5.28	Area 1 controlled generation response to external area event using TLB 4.	113
5.29	Normalized forecast and demand of parsed EIA data.	114
5.30	Changes in load caused by noise agent action during virtual wind ramp.	115
5.31	Area P_e and Load during morning peak simulation.	115

5.32	Area P_e and Load during morning peak simulation with load noise and governor deadbands.	115
5.33	System Frequency during morning peak simulation.	116
5.34	System Frequency during morning peak simulation with load noise and governor deadbands.	116
5.35	Calculated BA values during morning peak simulation.	117
5.36	Calculated BA values during morning peak simulation with load noise and governor deadbands.	117
5.37	System shunt bus voltage during morning peak simulation.	118
5.38	System shunt bus voltage during morning peak simulation with load noise and governor deadbands.	118
5.39	System shunt bus MVAR during morning peak simulation.	119
5.40	System shunt bus MVAR during morning peak simulation with load noise and governor deadbands.	119
5.41	Changes in load caused by noise agent action during virtual wind ramp. . .	120
5.42	Generator P_e distribution during virtual wind ramp under ideal conditions..	121
5.43	Generator P_e distribution during virtual wind ramp with load noise and governor deadbands.	121
5.44	System frequency response to virtual wind ramp under ideal conditions. . .	122
5.45	System frequency response to virtual wind ramp with load noise and governor deadbands.	122
5.46	Calculated BA values during virtual wind ramp under ideal conditions. . .	122
5.47	Calculated BA values during virtual wind ramp with load noise and governor deadbands.	123
5.48	Area P_e and P_{load} during virtual wind ramp under ideal conditions.	123
5.49	Area P_e and P_{load} during virtual wind ramp with load noise and governor deadbands.	123
5.50	System shunt bus voltage during virtual wind ramp under ideal conditions..	124
5.51	System shunt bus voltage during virtual wind ramp with load noise and governor deadbands.	124
5.52	System shunt bus MVAR during virtual wind ramp under ideal conditions..	125
5.53	System shunt bus MVAR during virtual wind ramp with load noise and governor deadbands.	125
5.54	Long-term dynamic settings for feed-forward governor simulation.	127
5.55	Feed-forward governor frequency comparison.	128
5.56	Feed-forward governor electric power comparison.	128

5.57	Long-term dynamic settings for variable system inertia simulation.	129
5.58	Frequency effects of system damping.	130
5.59	Frequency effects of varied system inertia.	130
5.60	Governor response to varied system inertia.	131
5.61	Varied system inertia during simulation.	131
5.62	System frequency response to varying system inertia.	131
A.1	Dyd file used in six machine validations.	143
B.1	An example of a full .py simulation file.	145
B.2	Required .py file for external AGC event with conditional ACE.	146
B.3	Required .ltd file for external AGC event with conditional ACE.	147
B.4	Required .ltd file for forecast demand scenario with noise and deadbands. . .	151
D.1	Area 1 valve travel using no deadband.	154
D.2	Area 2 valve travel using no deadband.	154
D.3	Area 3 valve travel using no deadband.	154
D.4	Area 1 valve travel using a step deadband.	155
D.5	Area 2 valve travel using a step deadband.	155
D.6	Area 3 valve travel using a step deadband.	155
D.7	Area 1 valve travel using a no-step deadband.	156
D.8	Area 2 valve travel using a no-step deadband.	156
D.9	Area 3 valve travel using a no-step deadband.	156
D.10	Area 1 valve travel using a non-linear droop deadband.	157
D.11	Area 2 valve travel using a non-linear droop deadband.	157
D.12	Area 3 valve travel using a non-linear droop deadband.	157
D.13	Area 1 valve travel in a non-homogeneous deadband system.	158
D.14	Area 2 valve travel in a non-homogeneous deadband system.	158
D.15	Area 3 valve travel in a non-homogeneous deadband system.	158
E.1	Frequency response to generation loss event in area 2.	159
E.2	Calculated values during generation loss event in area 2.	159
E.3	Area 1 controlled generation response to generation loss event in area 2. . .	159
E.4	Area 2 controlled generation response to generation loss event in area 2 . . .	160
E.5	AGC frequency response to area 2 base case scenario.	160
E.6	Calculated BA values during area 2 AGC tuning.	160
E.7	Area 1 controlled generation response during area 2 AGC tuning.	161
E.8	Area 2 controlled generation response during area 2 AGC tuning.	161
E.9	AGC frequency response with noise and deadbands.	161
E.10	Calculated BA values with noise and deadbands.	162

E.11	Area 1 controlled generation response to noise and deadbands.	162
E.12	Area 2 controlled generation response to noise and deadbands.	162
E.13	Frequency response to event using TLB 0.	163
E.14	Calculated BA values during an even using TLB 0.	163
E.15	Area 1 controlled generation response to external area event using TLB 0. .	163
E.16	Area 2 controlled generation response to internal area event using TLB 0. .	164
E.17	Frequency response to event using TLB 4.	164
E.18	Calculated BA values during an event using TLB 4.	164
E.19	Area 1 controlled generation response to external area event using TLB 4. .	165
E.20	Area 2 controlled generation response to internal area event using TLB 4. .	165

List of Equations

2.1	Branch Current Flow	9
2.2	Branch Real Power Flow	9
2.3	Branch Reactive Power Flow	9
2.4	Per-Unit Swing Equation	10
2.5	Per-Unit Speed Deviation	10
2.6	Area Control Error	12
2.7	Area Frequency Response Characteristic	12
4.1	Total System Inertia	21
4.2	System Accelerating Power	21
4.3	Combined Swing Equation	21
4.4	Distribution of Accelerating Power	22
4.5	Random Noise Injection	30
4.6	Data for Difference Plot	47
4.7	Absolute Average	47
4.8	Percent Difference	47
4.9	Weighted Frequency	48
5.1	Balancing Authority ACE Limit	95
5.2	Frequency Trigger Limit	95
5.3	Variable Frequency Bias	100

Glossary of Terms

Term	Definition
ABM	Agent Based Modeling
ABS	Agent Based Simulation
AC	Alternating Current
ACE	Area Control Error
AGC	Automatic Generation Control
AMQP	Advanced Message Queue Protocol
API	Application Programming Interface
BA	Balancing Authority
BES	Bulk Electrical System
CLR	Common Language Runtime
CTS	Classical Transient Stability
DACE	Distributed ACE
DTC	Definite Time Controller
EIA	United States Energy Information Administration
ERCOT	Electric Reliability Council of Texas
FERC	Federal Energy Regulatory Commission
FTL	Frequency Trigger Limit
GE	General Electric
Hz	Hertz, cycles per second
IACE	Integral of ACE
IPC	Interprocess Communication
IPY	IronPython
ISO	Independent Service Operator
J	Joule, Neton meters, Watt seconds
LFC	Load Frequency Control
LTD	Long-Term Dynamic
NERC	North American Electric Reliability Corporation
ODE	Ordinary Differential Equation

Term	Definition
P	Real Power
PI	Proportional and Integral
PMIO	PSLF Model Information Object
PSDS	PSLF Dynamic subsystem.
PSLF	Positive Sequence Load Flow
PSLTDSim	Power System Long-Term Dynamic Simulator
PSS	Power System Stabilizer
PST	Power System Toolbox
PU	Per Unit
PY3	Python version 3.x
PyPI	Python Package Index
Q	Reactive power
RACE	Reported ACE
RTO	Regional Transmission Organization
SACE	Smoothed ACE
SI	International System of Units
TLB	Tie-Line Bias
TSPF	Time-Sequenced Power Flow
US	United States of America
VAR	Volt Amps Reactive
W	Watt, Joules per second
WECC	Western Electricity Coordinating Council

1 Introduction

Over the past 140 years, an increasing demand for electricity has forced power systems to evolve. The earliest power generating stations designed to meet local needs evolved to many modern remote units to serve a wide area of distributed customers. Power systems have continuously gotten larger and more complicated. As most things do, these changes came about quite naturally. Additional generation stations were built and connected together to meet the ever increasing electric demand. Areas of demand became more dispersed and the use of high voltage transmission lines were used to link nearby power systems together.

The gradual connection of many remote power systems not only allowed for a pooling of resources, but with appropriate control, also offered increased system stability and reliability. Unfortunately, independent systems developed non-uniformly and created a complex network that requires intricate control and near constant attention. To further muddle matters, the aging North American electric power system is often pushed to its operating limits. This is due in part to, permitting or financial complications that may arise when attempting to add or update infrastructure to relieve system stress. Approved additions must be properly planned to handle a variety of environmental, economic, and social concerns. Well planned system modifications contribute long-term operational benefits while adhering to federal mandates. As such, to simply avoid complicated new construction, various control schemes have been employed to better manage existing power system assets. However, advanced control requires advanced planning and testing.

Fortunately, we live in the future and can use computers to monitor, control, design, and simulate power systems in ways not previously possible. Networked system monitoring has enabled digital controls to act on remote data and provided a way to create time-stamped histories of events as they unfold. Computer software can help analyze system characteristics and predict issues arising from common contingencies. Dynamic computer simulations can further the study of various events. Typical dynamic simulation focus is on system reactions tens of seconds after an event; while realistic recovery may require multiple minutes.

The engineering goal of this work is to simulate various frequency and voltage conditions that represent ‘long-term’ engineering problems, such as system recovery, daily load cycles, or wind ramp response, and provide customizable controls that can be used to simulate user control systems or simply explore possible system responses. A long-term dynamic simulation framework based on a time-sequenced power flow technique is believed to be capable of accomplishing such a task. Furthermore, the resulting open-source software may be expanded upon by future research for other related applications.

A simulation tool to accomplish the desired engineering goal does not yet exist. Therefore, a time-sequenced power flow simulation software program must be created and validated. Certain elements such as system model format and power-flow solver can be reused from previously existing software solutions, but features like automatic generation control, programmable logic controllers, and dynamic governor models must be created. Due to time limitations and engineering intuition, several modeling simplifications and assumptions are expected to be employed.

Specific requirements have been identified to facilitate software development and ensure the end product can accomplish stated goals. General code structure must allow for modular expansion, custom procedure insertion, and large system scaling. The software must also be able to run full base case scenarios involving thousands of buses and generators. The finished code solution must be able to output data that can be validated against industry standard software. Output data should provide enough detail so that analysis of system control validity and efficiency can be performed.

This thesis provides a brief explanation of some basic concepts involved with the electrical engineering aspects of this work, as well as the computer science, or software aspects. The created software is explained and validated before engineering applications are introduced. Simulation results related to engineering applications are then presented and analyzed. Finally, conclusions are stated and future work suggested.

2 Electrical Engineering Background

2.1 Power System Basics

Before covering the physical structure of power systems, it is worth providing a general introduction to the regulatory and operational structure of the North American bulk electric system (BES). The BES, more commonly known as ‘the grid’, is regulated by multiple tiers of organizations. At the top of the United States (US) bureaucratic pyramid is the federally empowered Federal Energy Regulatory Commission (FERC), which regulates natural gas and hydroelectric power projects as well as the interstate transmission of natural gas, oil, and electricity [19]. With authority granted by FERC, the North American Electric Reliability Corporation (NERC), aims to assure the effective and efficient reduction of risks to the reliability and security of the BES [44]. NERC also develops and enforces reliability standards and provides training to industry personnel. Under FERC and NERC, exist a collection of regional transmission organizations (RTOs), independent service operators (ISOs), balancing authorities (BAs) and utilities. All entities have a specific purpose and area of operation. This research focuses on BA action. In general, BAs are responsible for balancing supply and demand in defined areas of a power system while adhering to federal standards [9].

In the most basic physical sense, a power system includes a source of electric generation connected to some kind of electric consumption, or load. Modern alternating current (AC) power systems often resemble Figure 2.1.

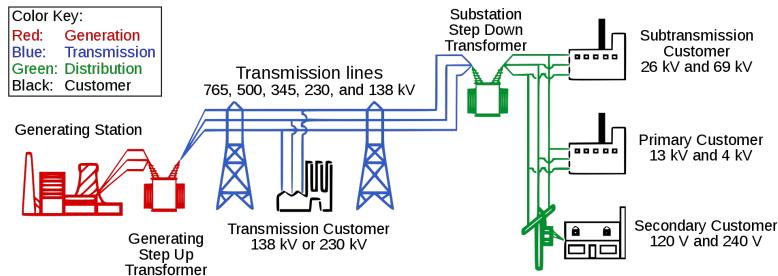


Figure 2.1: General form of a modern power system [30].

Power system loads are generally located wherever people feel the need to use electric-

ity. Generation is often placed in areas that have convenient natural resources nearby. This typically means that generation is built many miles away from large cities. Transformers and high-voltage transmission lines are used for efficient transfer of electric power over long distances. Figure 2.2 shows the location and relative size of electric generation in the US as of July 2019.

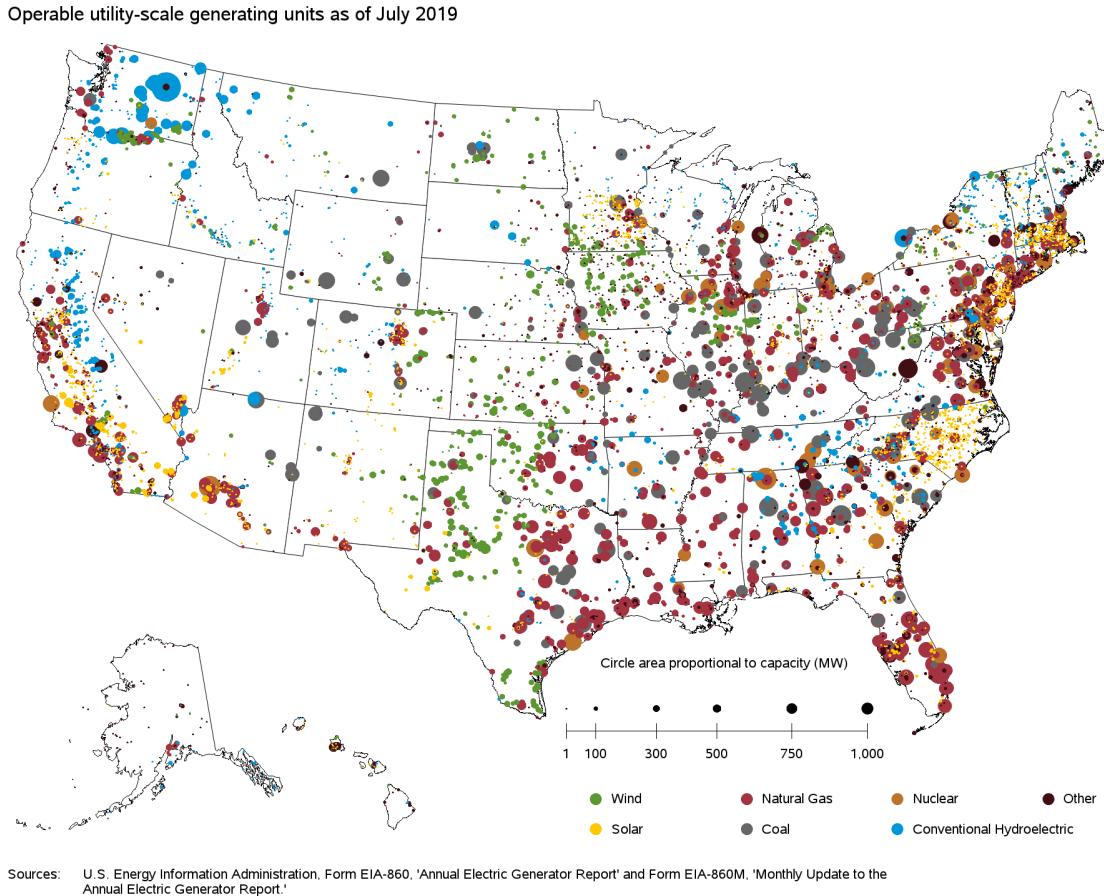


Figure 2.2: Location and size of US electric generation [10].

As expected, the northwest has large amounts of hydroelectric generation along major rivers. Solar generation is primarily located in southern states such as California and North Carolina. Wind generation can be found in and north of Texas towards the Great Lakes. Natural gas and coal are widely available in the eastern US. For interested Montanans, the four coal burning, steam generating, units at Colstrip are the large grey dot in the southeast corner of the state representing a nameplate capacity of 2,094 MW [14].

To help maintain stability, the North American grid is separated into interconnections, which can be thought of as relative frequency zones. This means that the system frequency and phase can be different between two interconnections, but must be synchronous inside the interconnection. Figure 2.3 depicts the Western, Eastern, and Electric Reliability Counsel of Texas (ERCOT) interconnections that make up the US grid.

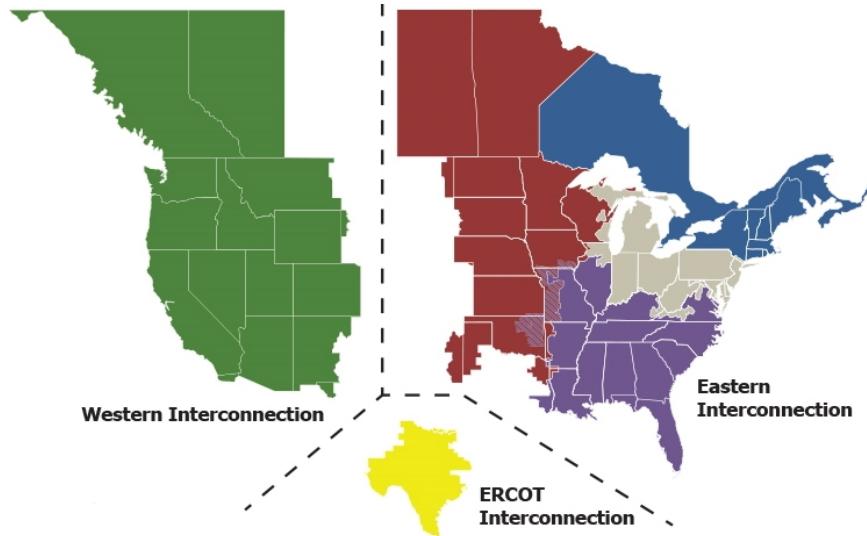


Figure 2.3: US Interconnections modified from [45].

The three main US interconnections are separated and distinguished by AC-DC-AC ties. The AC-DC-AC ties allow for a separation of frequency between interconnections. This is accomplished through a rectification and inverting process. Frequency content is removed when an AC signal is rectified into a DC signal. A new AC signal with any frequency and phase may be created through the inverting process. This rectification and inversion process allows power to be transported between the interconnections with acceptable losses.

The western interconnect, which includes most of Montana, has reliability standards that are created, monitored, and enforced by the Western Electricity Coordinating Council (WECC). A computer model of western interconnect, often referred to as the WECC, has been in use for over 20 years. As with any digital document that has been around for such a period of time, the model has become very large and some might say ‘cluttered’. The WECC model contains various unused, or outdated, information which makes use of the

model difficult. Further details on the WECC model are presented in Section 4.4.5.

In a well designed and operated power system, electric demand is always met. However, demand is always changing. To this end, automatic controls have been developed to better manage certain aspects of the grid. Electrical frequency, synonymous with generator speed, is often used as an input to such automatic controls. There is a direct correlation between frequency and the balance of electrical supply and demand. As Figure 2.4 so artfully depicts, if supply and demand are balanced, frequency remains static. If there is more demand than supply, frequency decreases. If the opposite is true, more supply than demand, frequency increases. Figure 2.4 illustrates that generated power must equal not only load, but also account for losses and any inter-regional interchange that may occur.

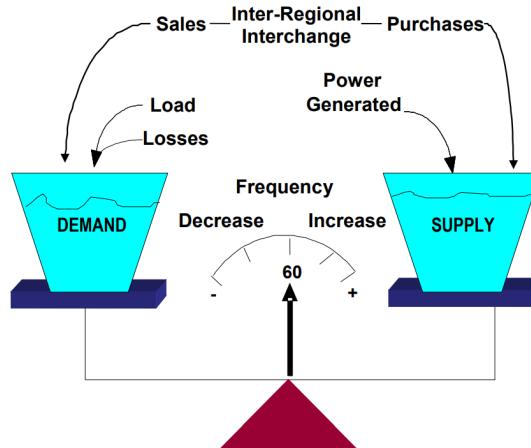


Figure 2.4: Frequency relation to electric supply and demand [49].

Most automatic control types within this thesis are distinguished by the time frame in which they operate. Figure 2.5 provides a classic frequency response to a loss of generation event with time scale classifications of automatic control responses.

The first automatic control, referred to as primary control, responds immediately after an event or perturbation and operates for tens of seconds. Primary control consists of turbine speed governors that act on deviations in system frequency from the nominal operating condition to stop frequency decline. Primary control occurs mostly during the arresting and rebound periods shown in Figure 2.5. Secondary control acts over tens of minutes during the

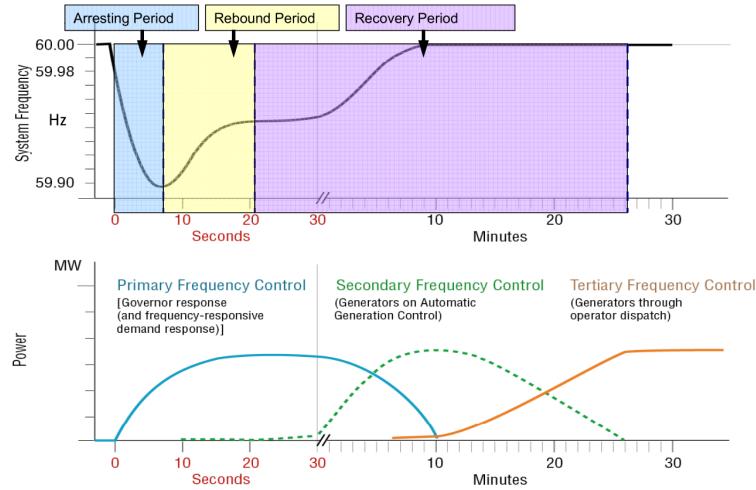


Figure 2.5: Classic frequency and automatic control response [6].

recovery period. Secondary control is referred to as automatic generation control (AGC) or load frequency control (LFC). AGC and LFC are often used interchangeably throughout the literature. This thesis uses AGC to refer to secondary control. AGC is configured by a BA to manage its area control error (ACE) which is a combination of interchange and frequency error. More information on AGC is presented in Sections 2.5 and 5.3.3. Additional operator defined automatic controls, such as relaying, tap changing, and plant level control, may be classified into either time category based on defined settings.

System stability and reliability are of utmost importance in the electric power game. Most Americans rely on electricity for daily activities of widely varying consequence. In the simple economic view, if an electric system does not work, customers can not buy electricity, and there is absolutely no chance to make a profit. Power outages, or other system misoperations, can be quite costly not only in a monetary sense, but can also result in the loss of life [26], [36]–[38], [59]. As such, simulation of a power system is required for planning additions or alterations. Simulations can be used to test and fine tune operational procedures like AGC settings or shunt switching schemes to ensure safe, economic, reliable, and correct operation. Power system simulation can also be used to explore events that may not be fully understood or occur rarely, but still require planned control measures.

2.2 Power Flow

The International System of Units (SI) measure for electric power is the watt W and has units of Joules/Second J/s. Watts can be thought of as the rate at which electrical work is performed, or the rate of energy transfer over time. Utility companies often charge for electric power by the amount of kW used per hour, or kilowatt hour kW h ($3.6E6$ J). For example, operating a 650 W computer power supply for one hour would require 0.65 kW h ($2.34E3$ J).

Most electric customers are only responsible for real, or active, power usage and are unaware of reactive power entirely. To simply state the difference between real power and reactive power: real power can be thought of as energy that does actual work, such as moving a conveyor belt, crushing rocks, or heating an element; reactive power can be thought of as energy spent creating and maintaining electric and/or magnetic fields. In large power systems, supply and demand is often reported in terms of megawatts MW for real power P, and MVAR for reactive power Q.

In an AC system, the quantity of power flowing on a line (also referred to as a branch), between two buses can be found by first calculating the line current. Equation 2.1 shows that the line impedance $R+jX$, voltage magnitude V_x , and voltage angle δ_x of two connected buses are required to calculate line current I. For example, if a system has any two buses that can be related to Figure 2.6, the equations to calculate current, real power, and reactive power flow between them is shown in Equations 2.1-2.3 respectively.

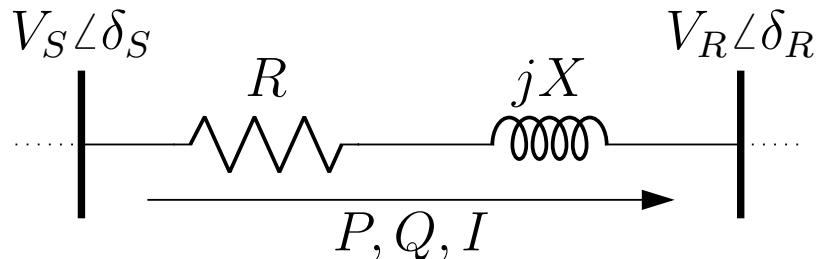


Figure 2.6: Two buses with power flow between them.

$$I = \frac{V_S e^{j\delta_S} - V_R e^{j\delta_R}}{\sqrt{3}(R + jX)} \quad (2.1)$$

$$P = \sqrt{3}V_S|I| \cos(\delta_S - \angle I) \quad (2.2)$$

$$Q = \sqrt{3}V_S|I| \sin(\delta_S - \angle I) \quad (2.3)$$

It should be noted that the above equations are per-unit (PU) and the square root of three is used to handle per phase and line-to-line value transforms. In practice, and thus simulation, all calculated values may prove useful. Megawatts on a line is a common way to describe real power flow, transmission lines and other equipment may be rated in current, and MVAR flow can be useful in shunt switching schemes to control bus voltage.

In reality, power flow is determined by physical measurements. In simulation, system characteristics are modeled and certain bus values are calculated using a power-flow computer program. The results from such a program, referred to as a power-flow solution, provides a steady state system operating condition based on certain known values. There are many methods to solve a power flow as [25] and [35] show, but detailed explanation is beyond the scope of this thesis. Suffice it to say that the power-flow problem involves system of non-linear equations that compute the voltage magnitude and phase angle at each bus in a power system under balanced three-phase steady state conditions [25]. Solutions often involve iteration until a set tolerance of ‘mis-match’ from an expected value is reached. Once a power-flow solution is achieved, Equations 2.1-2.3 can be used to find equipment loadings and line power flow anywhere in a system.

2.3 The Swing Equation

The swing equation is the basis of dynamic generator speed and is derived using Newton's second law. The equation got its name because it describes the swing in rotor angle during a disturbance [35]. Generator speed can be described by changes in rotor angle from a known reference. In the case of electric generators, frequency is directly related to turbine speed and either word is often used interchangeably.

The PU swing equation, Equation 2.4, shows how acceleration of generator frequency $\dot{\omega}$ is directly related to the balance between mechanical power P_{mech} supplied by a generator, and electrical power P_{elec} required by a load.

$$\dot{\omega} = \frac{1}{2H} \left(\frac{P_{mech} - P_{elec}}{\omega} - D\Delta\omega \right) \quad (2.4)$$

It can be seen that machine inertia H has an inverse relationship to the magnitude of $\dot{\omega}$. The D term may be used to represent damping forces any time a generator deviates from synchronous speed [25] however, it is often assumed to be zero. Per-unit speed deviation,

$$\Delta\omega = \omega_{rated} - \omega, \quad (2.5)$$

is a simple calculation used to scale the damping term and used in other to-be-described applications. The nature of PU equations dictate that rated speed be equal to one. Historically, the current frequency ω was often ignored, or assumed to be equal to one. This decision was to make calculations easier since speed deviation in PU is often very small. Modern usage of the swing equation typically accounts for frequency effects.

In a steady state operating condition, P_{mech} is equal to P_{elec} and angular acceleration $\dot{\omega}$ is equal to zero. If there is an increase in required P_{elec} , then $\dot{\omega}$ is negative and rotor angle (generator frequency) declines. If P_{mech} is larger than P_{elec} , $\dot{\omega}$ is positive and frequency increases. Detailed derivations and discussions on the swing equation can be found in [2], [25], [35] or most any decent book related to power system dynamics.

2.4 Turbine Speed Governors

Turbine speed governors, often just governors, act to arrest frequency change by adjusting a generator's power reference setting. Governors are classified as primary control because of their typically fast response. While many detailed models of governors exist, almost all include a permanent droop or regulation constant R . The droop constant is a ratio of change in frequency to change in power. R is often written in percent or a decimal representing a percent and assumed to be negative. Figure 2.7 shows how R , the slope of the diagonal line, affects the resulting $\Delta\omega$ caused by change in power ΔP .

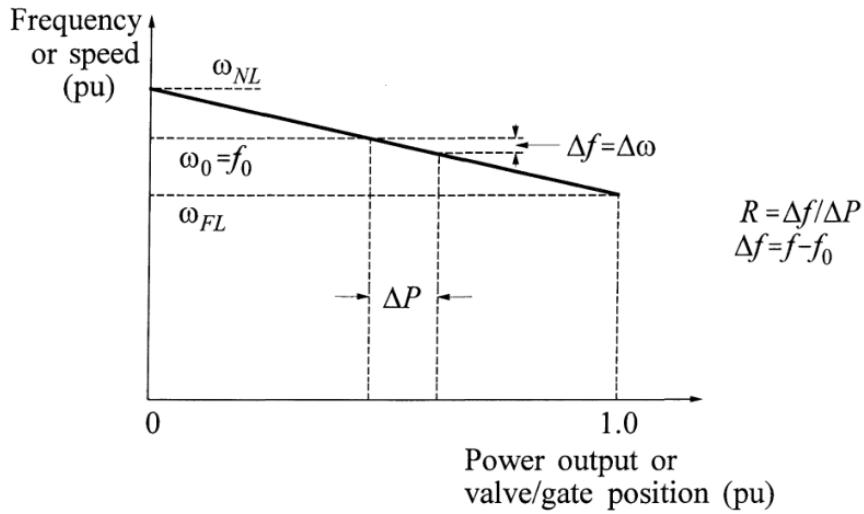


Figure 2.7: Ideal governor droop action [35].

Droop is a PU value so if a generator has a droop of 0.05, or 5%, then a 5% change in frequency would cause a 100% change in mechanical power output. Governor settings are dictated by FERC while NERC offers interconnection suggestions. The most common R is 5% for nearly all types of generators [50].

The input to a turbine speed governor is system frequency measured in Hertz Hz and reference power set point in MW. The output can be thought of as supplied mechanical power in MW. To prevent unnecessary control action, inputs to governors often have deadbands and may be delayed or filtered. Additional information on governor control is presented in Section 5.3.

2.5 Automatic Generation Control

Automatic generation control (AGC) is a form of secondary control that works to correct system frequency and area interchange error. AGC does this by calculating an area control error (ACE) that is then distributed to generation units under AGC control. The received AGC signals are used to adjust the generators mechanical power reference point. How AGC is distributed varies according to operator discretion so that operating or economic requirements are met.

NERC standards related to the management of ACE and requires ACE to be reported in a specific way [48]. As shown in Equation 2.6, reporting ACE (RACE) is composed of a variety of terms that may be relevant to electric operators.

$$RACE = (NI_A - NI_S) - 10B(F_A - F_S) - I_{ME} + I_{ATEC} \quad (2.6)$$

The difference between scheduled net interchange NI_S and actual net interchange NI_A is measured in MW. I_{ME} and I_{ATEC} , representing interchange meter error and area time error correction respectively, are also measured in MW. The difference between actual frequency F_A and scheduled frequency F_S is scaled by a frequency bias term $10B$ so that the result is also in MW. The 10 is required because B has units of MW/0.1Hz. By convention, positive $RACE$ indicates a general over-generation condition and negative $RACE$ is representative of an under-generation situation. Section 5.3.3 provides more details on AGC action.

For clarification, frequency bias B is not the same as area frequency response β [49]. B is an approximation of β used in ACE calculations and AGC action, while β is used to predict an area's response to perturbances. Equation 2.7 shows how β for an area may be calculated with N generators under droop control [25].

$$\beta = \sum_i^N \frac{1}{R_i} \quad (2.7)$$

2.6 Reactive Power and Voltage Control

While AGC handles frequency and area interchange control, system voltage and reactive power must also be controlled. Various objectives must be met to maintain an efficient and reliable power system [35]. Bus voltage must be held near a known scheduled value so that system components operate in a predictable and safe way. Reactive power flow should be minimized to reduce reactive losses. Depending on loading conditions and generator VAR output, bus voltages and reactive power flow vary. A lack of available reactive power in a system can lead to lower bus voltages. Unchecked, declining voltages may lead to voltage collapse which can then lead to chain reactions of automatic tripping.

Voltage, unlike system frequency, is a local issue and must be resolved using local power system devices. Low voltage situations may be resolved by the addition of capacitive VARs and high voltage situations call for the removal of capacitive VARs. Alternatively, Inductive VARs may be used in the opposite manner. Generators whose primary purpose is to produce reactive power are called synchronous condensers. Due to the nature of some renewable energies, areas with large amounts of renewable generation may not be able to produce adequate VARs for voltage control. Thus, to meet operational objectives, synchronous generator and switchable shunt control becomes an important consideration in modern automatic control configurations.

3 Software Background

3.1 Classical Transient Stability Simulation

Classical transient stability (CTS) simulation is a simulation approach commonly used to test a power system's ability to remain stable after a large step perturbation such as a generator trip. The time frame CTS focuses on is milliseconds to tens of seconds, and as such, requires time steps of milliseconds. While this is an appropriate approach for relatively short periods of time, complicated modeling issues may lead to questionable results over the course of longer simulations.

General Electric's (GE's) Positive Sequence Load Flow (PSLF) is an industry standard power-flow solver and CTS simulation program. The continued development of PSLF has produced a large library of dynamic models and components for use in CTS simulation. PSLFs dynamic library has enabled a wide variety of system models to be created. For example, multiple full WECC base cases have been modeled in PSLF over the past 20 years and are continuously being updated. Certain versions of PSLF offer a Python 3 application programming interface (API) and a .NET API. Despite each API being at various levels of development and functionality, both offer a modern way to communicate with PSLF.

3.2 Python

Python is an interpreted high-level general purpose programming language that utilizes object oriented techniques and emphasizes code readability [34]. Guido Van Rossum first implemented a version of Python in December of 1989 [55]. Python is freely available and distributable for multiple computing platforms [52].

3.2.1 Python Packages

Software modules that expand the functionality of Python are referred to as packages and are freely available from the Python Package Index (PyPI). This work utilizes multiple packages to varying degrees, though SciPy and Pika are among the most heavily used. SciPy

is a collection of packages that include NumPy for numerical computing and Matplotlib for MATLAB style plotting [58]. Additionally, `scipy.signal.lsim` was used to solve state-space equations that are common in electrical engineering dynamics. To avoid rewriting well known integration routines, `scipy.integrate.solve_ivp` was used to perform Runge-Kutta integration of the swing equation to find system frequency. The Python implementation of the advanced message queuing protocol (AMQP) used for this project was Pika [20].

3.2.2 Varieties of Python

Python has gone through numerous versions over the course of development and has been ported and adapted according to need. IronPython (IPY) is an open-source .NET compatible version of Python written in C# [1] and is able to use the common language runtime (CLR) package required for properly interacting with the GE PSLF .NET library. As such IPY, more specifically 32-bit IPY, is used to interface with the PSLF .NET library. However, the most current stable IPY release is based off of Python 2.x and can only use Python packages compatible with 2.x. Python 3 (PY3) is ‘the future of Python’ and has many more maintained and useful community created packages. Some packages area available for both Python version 2 and 3, but many are not. As of this writing, the current version of Python is 3.8. A majority of project code was written using Python 3.7, but should continue to be compatible with future versions of PY3.

3.2.3 Python Specific Data Types

Python has various common data types found in other programing languages such as integer, string, list, and float. Python also has some unique data types. One unique data type is called a *dictionary* which is a collection of key value pairs. Dictionary keys are strings, and the value can be any other data type, including a dictionary. A benefit of using a dictionary is that it doesn’t matter ‘where’ in an object a certain data point is located, only that it exists somewhere in the object. The key value pair eliminates the need to focus on indexing lists or arrays as other programming languages may require. Additionally, native python methods allow for simple searching and iteration of dictionaries.

Another somewhat unique python data type is a tuple. Tuples are essentially the same as lists, except defined using parenthesis instead of brackets and the data inside can not be changed in any way. While this may seem like a hindrance, it creates peace of mind when using tuples for important data references.

A powerful characteristic of Python is that everything is an object, and variables act as references, or pointers, to data. These Pythonic points were relied upon heavily during software development to make large collections of objects and references.

3.3 Advanced Message Queueing Protocol

Advanced message queueing protocol (AMQP) is a software messaging protocol that can be used for interprocess communication (IPC). The specific application of AMQP in this case was to enable a PY3 process to communicate with an IPY process on a Windows based machine. The idea behind AMQP is that of a virtual 'broker' which receives messages from various processes and places them into specific named queues. The named queues are accessed by other processes that check for, and receive, any queued messages.

It should be noted that Erlang was required to allow use of RabbitMQ, and the PY3 and IPY Pika packages were required so that Python could use AMQP. While detailed descriptions of these softwares is beyond the scope of this research: Erlang is an open source programming language and runtime environment, RabbitMQ is an open source AMQP broker software, and Pika is a Python package that works with RabbitMQ. The correct installation of these software packages is necessary for the created research software to function.

3.4 Agent Based Modeling

Agent Based Modeling (ABM), or Agent Based Simulation (ABS), is oriented around the idea that any situation can be described by agents in an environment, and a definition of agent-to-agent and agent-to-environment interactions [53]. The ABM coding style lends itself towards modular coding and natural expandability as each agent class may have inherited

methods and variable characteristics. As an example, [3] used an ABM approach to test the efficiency of differing airplane boarding methods. Passengers were treated as agents interacting with each other and the airplane environment. Each passenger agent may have had different characteristics, but performed a similar actions each simulation time step. ABM is applied to the coding of this project in that a power system acts as the environment, and all power system objects, such as buses, loads, and generators, are treated as agents.

Each time step, agents may perform their *step* method that executes code unique to the specific agent, but generally the same among agent types. For example, a timer agent step method may include checking a specified value and incrementing an accumulator according to a logic operation. If the accumulator becomes larger than a set threshold value, the timer may raise an activation flag. Agent action is meant to be direct and generic so that agents can be reused and nested inside other agents. Continuing the example, the timer agent may be nested inside another agent that checks the timers activation flags each step and takes action depending on the returned status. Actions can be arranged into a sequence of agent steps in a discrete time simulation, and then repeated ad infinitum. The idea of sequencing agent steps can be applied to systems of nearly any size and composition as long as agents have a step function and can reference other agents inside the environment.

4 Software Tool

The newly created software tool for this research was named PSLTDSim (Power System Long-Term Dynamic Simulator). Python was the code language of choice, and both PY3 and IPY processes are created during simulation. PSLF and AMQP also play integral roles in PSLTDSim. The basic idea behind PSLTDSim is to use time-sequenced power flows and external dynamic calculations to simulate long-term dynamic (LTD) power system behavior. Various engineering simplifications and assumptions are employed by PSLTDSim.

This chapter describes what time-sequenced power flows are, what PSLTDSim assumes or simplifies, and what PSLTDSim actually does. Software validations using PSLF simulation output as a reference are presented and discussed at the end of this chapter.

4.1 Time-Sequenced Power Flows

The left side of Figure 4.1 is a visual representation of a single power-flow solution. The power-flow solution can be thought of as a snapshot of steady state system operation. A period of steady state system operation could be imagined by repeating this single power-flow solution in a time sequence. Dynamic system behavior could be realized if small changes are made to the power-flow problem inputs and the ensuing power-flow solution converges. The right side of Figure 4.1 illustrates the idea of time-sequenced power flow (TSPF) as a collection of slightly changing power-flow solutions. The TSPF method for dynamic simulation involves performing dynamic modeling calculations outside of, or inbetween, each power-flow solution.

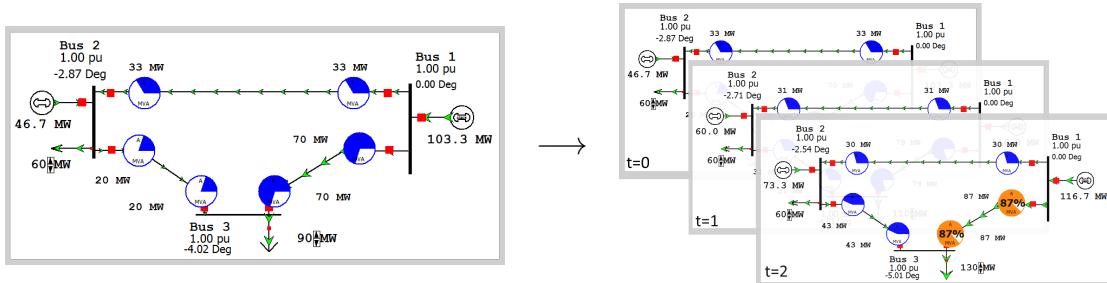


Figure 4.1: Time-sequenced power flows visualized.

Differences between TSPF and CTS simulation techniques stem from the time frame of simulation focus and differing dynamic calculations. TSPF is focused on long-term events that take place over the course of minutes to hours. CTS simulation focuses on events that are tens of seconds in duration. This difference in focus leads to each method utilizing a different appropriate time step. A TSPF time step may be 0.5 to 1 seconds while CTS uses sub-cycle time-steps in the millisecond range.

Calculations of CTS simulation include generator dynamics that are aggregated and simplified in TSPF. Each method starts with a power-flow solution, but CTS simulation performs back calculations to set initial states of various dynamic models. Future CTS states, and resulting system behaviors, are dictated by dynamic model interactions which may go unstable. TSPF also uses dynamic models, but updated values are sent to a power-flow solver; and the power-flow solution provides new steady state system values. These differences create output data that is of different resolutions and captures slightly different system characteristics. Figure 4.2 shows a comparison of CTS and TSPF data.

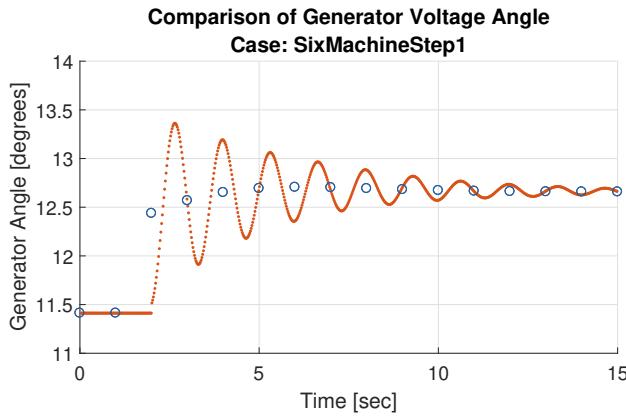


Figure 4.2: CTS and TSPF data output comparison.

The circles represent data from a TSPF simulation while dots show down-sampled data from PSLFs dynamic subsystem (PSDS) simulation. The two data sets are not the same during oscillatory behavior, but seem to match at steady state conditions. In the provided plot, TSPF data seems to follow the center of oscillation from the CTS simulation. This ‘oscillation averaging’ behavior was found to be common in TSPF results.

4.2 Simulation Assumptions and Simplifications

Numerous assumptions and simplifications were made due to fundamental differences between TSPF and CTS simulation. This section details such assumptions and simplifications and provides key TSPF equations.

4.2.1 General Assumptions and Simplifications

The focus of PSLTDSim is on the long-term operation of power systems. Since a power-flow solution is a stable steady state operating condition, it is assumed that the system of study is stable, and remains stable, for the entirety of the simulation. Other software packages should be used if transient stability is in question.

In general, system responses to large step type contingencies are not the focus of PSLTDSim. Instead, a more suitable event type is in the form of ramps, or repeated small step perturbances. Because of the more ‘gentle’ system scenarios involved with ramps, power system stabilizers (PSS) are not modeled. It is assumed that the system will not be disturbed enough to require meaningful PSS action.

Further assumptions include ideal generator excitors that can always maintain a given generator reference voltage. Modern excitors are assumed to be fast enough to hold reference voltages to small perturbances in the long-term. Despite the ideal reference voltage assumption, machine VAR limits **are** enforced according to model parameters. Should the need arise, future work can be done to incorporate any additional, or alternative modeling.

4.2.2 Time Step Assumptions and Simplifications

Multiple assumptions can be made concerning power system behavior when a time step of one second is used. The order of magnitude time step difference between CTS and TSPF allows models created for CTS simulations to be simplified for use in TSPF. Intermachine oscillations were ignored since subsynchronous resonances are sub-second and the time resolution of TSPF is not great enough to capture these phenomena. Additionally, in the long-term, these effects are minor when the system is stable. Without the need for

subsynchronous characteristics, generator modeling was greatly simplified. The only details required for a machine model are MW cap, machine MVA base, and machine inertia.

To further simplify generator modeling, all machines share a single system frequency that is calculated by a combined swing equation. The following two Sections explain how the combined swing equation is used in PSLTDSim. Governor models were also simplified. Section 4.2.5 explains the how PSLTDSim models governors.

4.2.3 Combined System Frequency

Instead of a frequency being calculated for each generator or bus, a single combined swing equation is used to model only one combined system frequency. This technique requires a known total system inertia H_{sys} and the total system acceleration power $P_{acc,sys}$. In a system with N generators, H_{sys} is calculated from each individual machines inertia as

$$H_{sys} = \sum_{i=1}^N H_{PU,i} M_{base,i}. \quad (4.1)$$

In a system with N generators, total system accelerating power is calculated as

$$P_{acc,sys} = \sum_{i=1}^N P_{m,i} - \sum_{i=1}^N P_{e,i} - \sum \Delta P_{pert}, \quad (4.2)$$

where $P_{m,i}$ is mechanical power and $P_{e,i}$ is electrical power of generator i and any system power injections, or perturbances, are accounted for in the $\sum \Delta P_{pert}$ term.

The combined swing equation, shown in Equation 4.3, uses $P_{acc,sys}$ and H_{sys} to calculate $\dot{\omega}_{sys}$. After integration, $\dot{\omega}_{sys}$ leads to the single combined system frequency ω_{sys} .

$$\dot{\omega}_{sys} = \frac{1}{2H_{sys}} \left(\frac{P_{acc,sys}}{\omega_{sys}} - D_{sys} \Delta \omega_{sys} \right) \quad (4.3)$$

For completeness, a damping term $D_{sys} \Delta \omega$ is included in Equation 4.3, but as Equation 2.4, D_{sys} is often set to zero while $\Delta \omega_{sys}$ is calculated using Equation 2.5 with ω_{sys} replacing ω .

4.2.4 Distribution of Accelerating Power

While system frequency can be calculated using total system accelerating power, to properly ‘seed’ the next power flow, each generator participating in the system inertial response must account for a portion of accelerating power absorption. The specific amount each generator absorbs is based on machine inertia. Equation 4.4 shows how the next electric power estimate $P_{e,EST,i}$ is created for generator i .

$$P_{e,EST,i} = P_{e,i} - P_{acc,sys} \left(\frac{H_i}{H_{sys}} \right) \quad (4.4)$$

Once all accelerating power is distributed to inertial responding generators, the new $P_{e,EST}$ value for each generator is used to seed a power flow. If the MW difference between resulting power supplied by the slack generator and estimated power output is larger than the set slack tolerance, the difference is redistributed as $P_{acc,sys}$ according to Equation 4.4 until slack tolerance is met, or a maximum number of iterations take place. Once the slack tolerance is met, the power-flow solution is accepted as the current state of the system under study.

4.2.5 Governor models

Long-term dynamic models do not require the detail of a full transient simulation model. For software validation purposes, a *tgov1* governor model was created as it appears in the PSLF documentation. This particular governor model was selected due to simplicity, and was later expanded upon to include an optional deadband and filtered input delay. The block diagram for the modified *tgov1* governor is shown in Figure 4.3. Blocks with a * next to them indicate they are optional, and only inserted into the model if user defined.

A slightly more generic governor was created based off the governor model used in Power System Toolbox (PST). This generic model, referred to as *genericGov*, is shown in Figure 4.4. The *genericGov* follows the time constant naming convention of the PST governor and includes the same optional blocks added to the modified *tgov1* model.

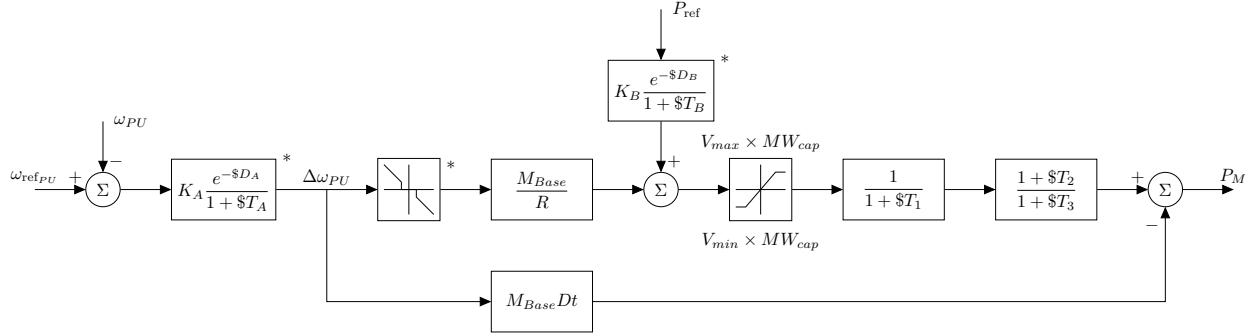


Figure 4.3: Block diagram of modified tgov1 model.

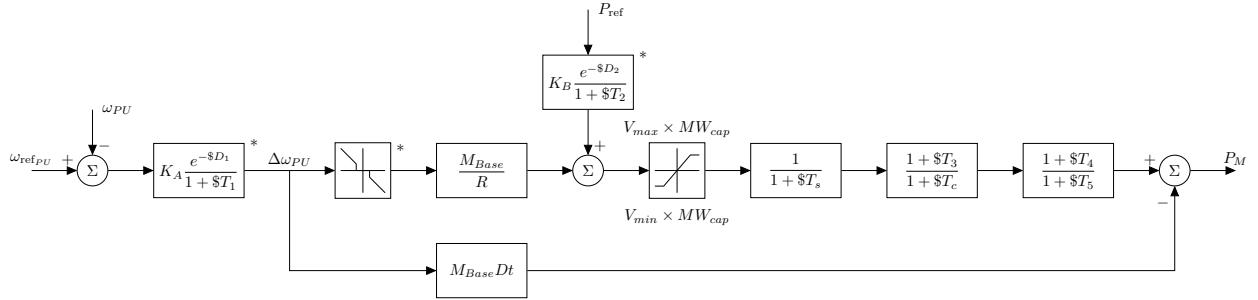


Figure 4.4: Block diagram of genericGov model.

4.2.5.1 Casting Process for genericGov

Modeling all PSDS governors would be a rather large task. Un-modeled governors encountered in a system dyd file were cast to a genericGov model based on assumed governor type. Table 4.1 shows the relation of PSDS model types to assumed governor setting type.

Table 4.1: Generic governor model casting between LTD and PSDS.

genericGov	Steam	Hydro	Gas
PSDS	ccbt1 gast w2301 ieeeg3 ieeeg1	g2wscc hyg3 hygov4 hygov hygovr pidgov	ggov1 ggov3 gpwscc

Universal governor settings, such as permanent droop and MW cap values are collected from the dyd file and used as R and MW_{cap} respectively. The particular time constants

for each model were selected according to typical settings associated with prime mover type that a PSDS model is assumed to represent. Table 4.2 lists the time constants used for each genericGov model type.

Table 4.2: Generic governor model parameters.

Parameter	Steam	Hydro	Gas
Ts	0.04	0.40	0.50
Tc	0.20	45.00	10.00
T3	0.00	5.00	4.00
T4	1.50	-1.00	0.00
T5	5.00	0.50	1.00

4.3 General Software Explanation

This section provides an explanation of the Power System Long-Term Dynamic Simulator (PSLTDSim). A flow chart showing a general overview of simulation action is shown in Figure 4.5. A simulation begins with user input of simulation specifications to PSLTDSim. The user input is then used to initialize the required simulation environment by PSLTDSim. The general simulation loop includes performing dynamic calculations and executing any perturbances which are then transferred to PSLF. A power-flow solution is then calculated by PSLF and relevant data sent back to PSLTDSim for logging. Various simulation variables are then checked to verify if the simulation loop is to continue or end. Once the simulation is complete, data is output and plots may be generated for the user to analyze.

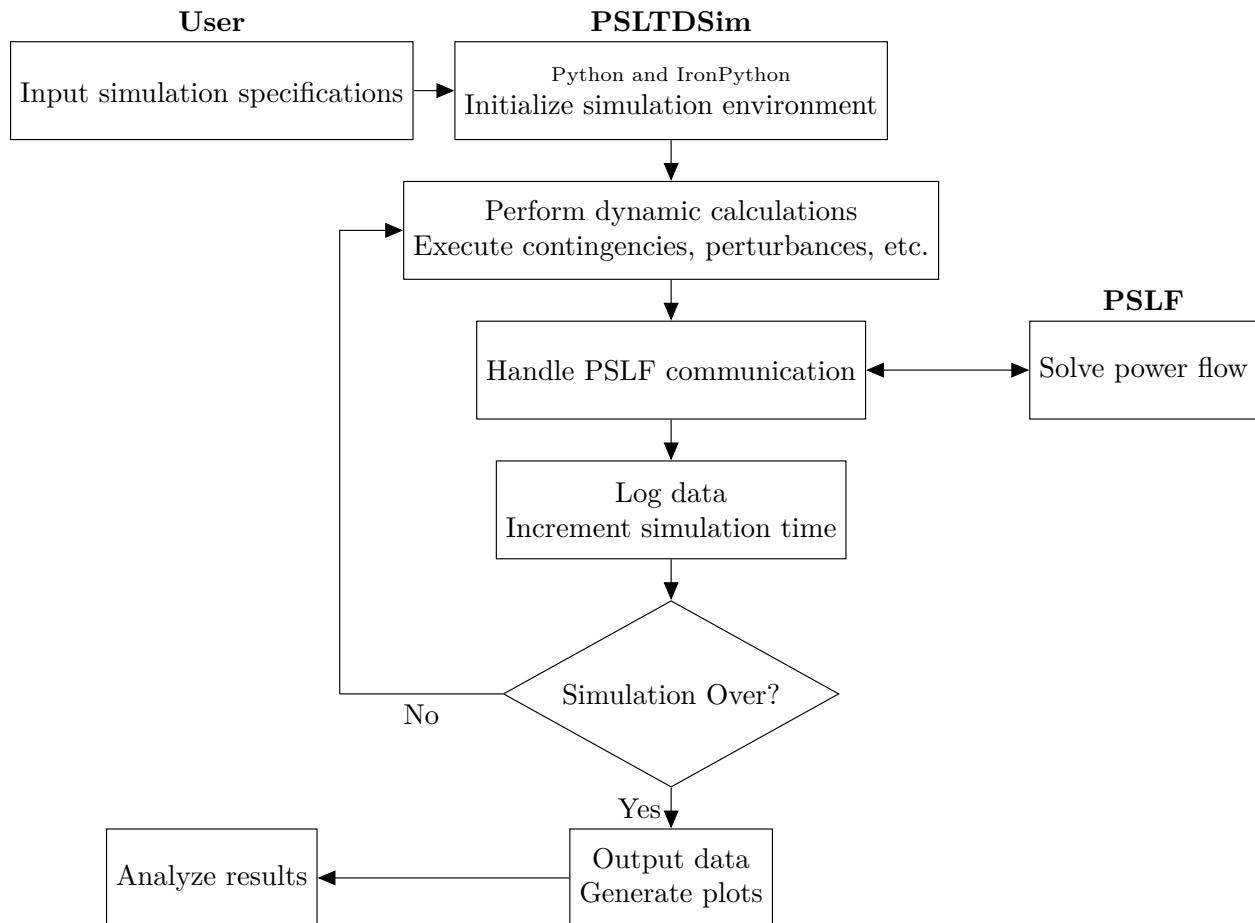


Figure 4.5: High level software flow chart.

4.3.1 Interprocess Communication

A process is another name for an instance of a running computer program. It is common for a computer program to run as a single process, however this is not always the case and not how PSLTDSim operates. Since processes are independent from each other, they do not share a memory space [22]. This effectively means that for two processes share data, some kind of interprocess communication (IPC) must be utilized.

Due to the current state of the GE Python 3 API, Ironpython (IPY) was required for functional PSLF software communication. Unfortunately, IPY is based on Python 2 and does not have packages necessary for numerical computation that Python 3 (PY3) possesses. The solution to these issues was to create a software that has an IPY process and PY3 process that communicate to each other via AMQP. The IPY process acts as a ‘middle man’ between PY3 and PSLF. Newly calculated dyanmive values from PY3 must be sent to PSLF via IPY, and after each new power-flow solution, PSLF values must be sent to PY3. This cycle continues as long as simulation is required. While the described AMQP solution has been shown to work, it is worth noting that message handling typically accounts for about one half of all simulation time, and sometimes more in larger cases.

4.3.2 Simulation Inputs

As with any simulation software, PSLTDSim requires specific inputs to operate correctly. Some required input is the same as that used by PSLF, while other input is Python based. Both types of inputs are described in this subsection.

4.3.2.1 PSLF Compatible Input

The power system model input used by PSLTDSim is the same .sav binary file used by, and generated from, PSLF. This is due the reliance of PSLTDSim on the power-flow solver included with PSLF. Python system dynamics are created based on the .dyd text files also used by PSLF. Information on creating .sav or .dyd files is beyond the scope of this text, but may be found in [23].

4.3.2.2 Simulation Parameter Input (.py)

Simulation parameter input is entered in a standard python .py file. Most simulation parameter input is collected in a Python dictionary named simParams. An example of the simParams dictionary defined inside the .py file is shown in Figure 4.6.

```

1  simParams = {
2      'timeStep': 1.0, # seconds
3      'endTime': 60.0*8, # seconds
4      'slackTol': 1, # MW
5      'PY3msgGroup' : 3, # number of Agent msgs per AMQP msg
6      'IPYmsgGroup' : 60, # number of Agent msgs per AMQP msg
7      'Hinput' : 0.0, # MW*sec of entire system, if !> 0.0, will be calculated in code
8      'Dsys' : 0.0, # Damping
9      'fBase' : 60.0, # System F base in Hertz
10     'freqEffects' : True, # w in swing equation will not be assumed 1 if true
11     'assumedV' : 'Vsched', # assumed voltage - either Vsched or Vinit
12     # Mathematical Options
13     'integrationMethod' : 'rk45',
14     # Data Export Parameters
15     'fileDirectory' : "\\\delme\\\200109-delayScenario1\\", # relative path from cwd
16     'fileName' : 'SixMachineDelayStep1', # Case name in plots
17     'exportFinalMirror': 1, # Export mirror with all data
18     'exportMat': 1, # if IPY: requires exportDict == 1 to work
19     'exportDict' : 0, # when using python 3 no need to export dicts.
20     'logBranch' : True,
21 }
```

Figure 4.6: An example of a simParams dictionary.

The simParams dictionary contains information required for the simulation to operate, such as time step, end time, base frequency, and slack tolerance. There are also parameters that alter how the simulation operates, such as integration method, inclusion of frequency effects, and how system inertia is calculated. Information related to data collection or export is also included in the simParams dictionary such as whether to log branch information or where the resultant data will be placed and what it will be named. Examples of valid dictionary keys, types of data, units of input, and a brief explanation is shown in Table C.2 located in Appendix C.

In addition to the simParams dictionary, simulation notes, absolute file paths to the desired .sav and .ltd.py file are also defined in this .py file. Dynamic input in the form of .dyd files are defined in a list to allow for using more than one .dyd file. The dynamic model overwriting that this feature was meant to incorporate has not been fully implemented as of this writing. An example of a valid simulation parameter input .py is shown in Appendix B as Figure B.1.

4.3.2.3 Long-Term Dynamic Input (.ltd.py)

The required .ltd.py file contains user defined objects related specifically to simulated events and control action. Further, this file is the ideal place for adding additional user input if the need should arise in future software development. Code in the .ltd.py file is standard Python and involves initializing objects attached to a *mirror* object. The mirror object is what PSLTDSim calls the total power system model. A description of the various objects that can be attached to the mirror is presented in the following sections. Most topics introduced are described completely, however, some options are more complex and described in later sections.

4.3.2.3.1 Perturbation List

The only list defined in the .ltd.py file is for entering simulation perturbances (often also referred to as contingencies or events). The list of single quoted strings describing system perturbances is defined as `mirror.sysPerturbances`. Common perturbances are changes in operating state and power, however, any value in the target agents current value dictionary may be changed. The format of the string is specific to agent type and perturbation, but strings follow the general format of 'Agent Identification : Perturbation Description'. Table 4.3 shows the format for the agent identification part of the perturbation string. Optional parameters are shown in brackets. If no ID is specified the first agent found with matching bus values will be chosen.

Table 4.4 describes the various parameters used to specify the action of the pertur-

Table 4.3: Perturbation agent identification options.

Agent Type	Identification Parameters	
load	Bus Number	[ID]
shunt	Bus Number	[ID]
gen	Bus Number	[ID]
branch	From Bus Number	To Bus Number [Circuit ID]

bance agent in the ‘Pertrubance Description’. The three valid options for the ‘Step Type’ and ‘Ramp Type’ field are **abs**, **rel**, and **per**. To make an absolute change to the new value, the **abs** type should be selected. To alter the target parameter by a relative value, the **rel** option should be used. If a percent change is desired, the **per** type should be used.

Table 4.4: Perturbation agent action options.

Type	Settings				
step	Target Parameter	Action Time	New Value	Step Type	
ramp	Target Parameter	Start Time	Ramp Duration	New Value	Ramp Type

Figure 4.7 shows various examples of valid perturbation agent definitions. Double quoted strings may be used to clarify perturbation descriptions. It should be noted that the target parameter is case sensitive. Additionally, if stepping a governed generator, both the mechanical power and power reference variables should be changed as shown in line 8 and 9 of Figure 4.7.

```

1 # Perturbation Examples
2 mirror.sysPerturbances = [
3     'gen 27 : step St 2 0',           # Set gen 27 status to 0 at t=2
4     'branch 7 8 2 : step St 10 0 abs', # Trip branch between bus 7 and 8 with ckID=2
5     'load 26 : ramp P 2 40 400 rel',   # ramp load 26 P up 400MW over t=2-42 seconds
6     'load 9 : "Type" ramp "Target" P "startTime" 2 "RAtime" 40 "RAval" -5 "RAtype" per',
7     'shunt 9 4 : step St 32 1',       # Step shunt id 4 on bus 9 on at t=32
8     'gen 62 : step Pm 2 -1500 rel',   # Step gen Pm down 1500 MW at t=2
9     'gen 62 : step Pref 2 -1500 rel', # Step gen Pref down 1500 MW at t=2
10    ]

```

Figure 4.7: Perturbation agent examples.

4.3.2.3.2 Noise Agent Attribute

A random noise agent was created to add random noise to all system loads. The noise agent is created in the .ltd.py file by defining the `mirror.NoiseAgent` attribute with the associated agent. Figure 4.8 shows an example of a noise agent being attached to a system mirror. The input arguments are: system mirror reference, percent noise to be added, a boolean value dictating random walk behavior, delay before noise is added, and the random number generator seed value.

```

1 # Noise Agent Creation
2 mirror.NoiseAgent = ltd.perturbation.LoadNoiseAgent(mirror, 0.05, walk=True, delay=0,
→ damping=0, seed=11)

```

Figure 4.8: Noise agent creation example.

At every time step, noise is injected into each load $P_{L,i}$ in the system according to

$$P_{L,i}(t) = P_{L,i}(t-1)[1 \pm N_Z Rand_i + D_{nz} \Delta\omega] \quad (4.5)$$

where N_Z represents the maximum amount of random noise to inject as a percent, $Rand_i$ is a randomly generated number between 0 and 1 inclusive, D_{nz} is the user input damping value, and $\Delta\omega$ is calculated according to Equation 2.5. The decision to add or subtract noise is

chosen by a unique randomly generated number. As described in [61], Equation 4.5 creates random walk behavior in load that is representative of real power systems. If random walk behavior is not desired, the walk input argument may be set to False and the scheduled load value is always used as $P_{L,i}(t - 1)$. Only mirror reference and percent of desired noise is required for noise agent creation. If no additional arguments are provided, default values for walk, delay, damping and seed are set to False, 0, 0, and 42 respectively.

4.3.2.3.3 Balancing Authority Dictionary

The mirror.sysBA dictionary is defined in the .ltd.py file and is used to configure BA agents. Individual BA dictionaries are nested inside the mirror.sysBA dictionary. The information entered in each nested dictionary describes how that particular BA acts on the specified area. Additional information on BA AGC options and action is presented in Section 5.3.3. Examples of BA parameterization are shown in Figures B.3 and B.4 of Appendix B. A description of each possible field is described in Table C.1 located in Appendix C.

4.3.2.3.4 Load Control Dictionary

To simplify changing all area loads according to a known demand schedule, a load control agent may be defined in the .ltd.py file. Similar to the balancing authority dictionary, each agent is defined as a named dictionary inside a the mirror.sysLoadControl dictionary. The load control agent requires area number, start time, time scale, and a list of tuples for demand information. Specific BA demand data can easily be acquired via the United States Energy Information Administration (EIA) website and saved as a .csv file. A script was written to parse and display demand changes over time as a relative percent change so that real load patterns could be applied to test systems of differing scale. An example of a single area load control agent is shown in Figure 4.9.

```

1 mirror.sysLoadControl = {
2     'testSystem' : {
3         'Area': 2,
4         'startTime' : 2,
5         'timeScale' : 10,
6         'rampType' : 'per', # relative percent change
7         # Data from: 12/11/2019 PACE
8         'demand' : [
9             #(time , Percent change from previous value)
10            (0, 0.000),
11            (1, 3.675),
12            (2, 6.474),
13            (3, 3.770),
14        ] , # end of demand tuple list
15    },# end of testSystem definition
16 }# end of sysLoadControl dictionary

```

Figure 4.9: Load control agent dictionary definition example.

The list of tuples named ‘demand’ defines the desired load change over time. The first value in each tuple is assumed to be a time value and the second number is a percent change value. The entered time value is scaled by the ‘timeScale’ value. It is assumed that both values in the first entry are always zero since there can be no relative change from negative time.

Relative percent ramps are used to alter load value between each entry. For example, if the load control agent shown in Figure 4.9 was used in a simulation, ramps would be created for each load in area 2 that increases load by 3.675% between time 2 and 10, 6.474% between time 10 and 20, and 3.770% between time 20 and 30. In total, a 100 MW load would be 114.548 MW after all ramps executed. The relative percent is based off the value of each load at start of each ramp. This method of relative percent changing allows for other perturbances, such as noise, to be applied to the same load without control conflicts. Alternative ramp types may be employed by changing the ‘rampType’ parameter, but are not created as of this writing. It should be noted that the first ramp starts at ‘startTime’, but all other ramps begin according to the calculated scaled time schedule. Additionally,

loads that are off at system initialization (status 0 at time 0) are ignored.

4.3.2.3.5 Generation Control Dictionary

To manipulate generation in the same way a load control agent manipulates load, a generation control agent may be defined in the .ltd.py file. The definition of a generation control agent is very similar to the definition of a load control agent by design, however, differences do exist. Generation control agents are defined in the dictionary named mirror.sysGenerationControl, have a list of strings detailing control generators, and have a time value tuple list named ‘forcast’. While the forecast misspelling was unintentional, and can be changed, time has proven it to be a minor detail. Other parameters inside the generation control agent definition (such as area, start time, time scale, and ramp type) function exactly the same as the load control agent. Forecast data is again collected from the EIA website and parsed in the same manner as demand data. Figure 4.10 shows an example of a generation control agent definition.

```

1  mirror.sysGenerationControl = {
2      'testSystem' : {
3          'Area': 2,
4          'startTime' : 2,
5          'timeScale' : 10,
6          'rampType' : 'per', # relative percent change
7          'CtrlGens': [
8              "gen 3 : 0.25",
9              "gen 4 : 0.75",
10             ],
11         # Data from: 12/11/2019 PACE
12         'forcast' : [
13             #(time , Precent change from previous value)
14             (0, 0.000),
15             (1, 5.137),
16             (2, 6.098),
17             (3, 4.471),
18             ],# end of forcast tuple list
19         }, #end of testSystem def
20     }# end of sysLoadControl dictionary

```

Figure 4.10: Generation control agent dictionary definition example.

The main difference between load and generation control agents is the addition of the ‘CtrlGens’ list of strings. Each string inside the ‘CtrlGens’ list is of the form: ”gen BusNumber ID : Participation Factor” where ID is optional. If ID is not defined, as shown in Figure 4.10, the first generator on the given bus will be controlled. The participation factor is used to distribute the total requested MW change in a more controlled fashion.

For example, if an area is generating 100 MW at time 0, the total requested area generation change by time 10 would be 5.137 MW. The generator on bus 3 would increase 1.28 MW while the generator on bus 4 would increase 3.85 MW. If a controlled generator has a governor, governor reference will be adjusted instead of mechanical power. Participation factor for all listed generators should always sum to 1.0 or improper distribution **will** occur. It should be noted that not all generation must be controlled for proper percent change of total area output power however, if a controlled machine hits a generation limit, excess changes are ignored.

Relative percent ramps are used to control generators in the same way as load so that a BA can also act on generators under generation control, however, this functionality is untested as of this writing.

4.3.2.3.6 Governor Input Delay and Filtering Dictionary

The inputs to modeled governors may be delayed, filtered, and gained using the Laplace domain block shown in Figure 4.11. If delay is not divisible by the simulation time step, rounding will occur.

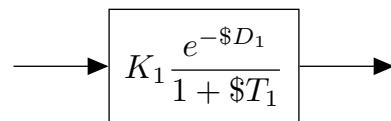


Figure 4.11: Block diagram of delay block.

To modify a governor model with a delay block, parameters may be entered in the

governor delay dictionary mirror.govDelay. Like previously described dictionaries, this is located in the .ltd.py file. Figure 4.12 shows an example of a valid delay dictionary that affects the governor of the generator on bus 3. Note that while genId is optional, if set to None the first found generator on the specified bus is used.

```

1 mirror.govDelay ={
2     'delaygen3' : {
3         'genBus' : 3,
4         'genId' : None, # optional
5         # (delay parameter, filter time constant, optional gain)
6         'wDelay' : (40,30),
7         'PrefDelay' : (10, 0),
8     }, # end of 'delaygen3' definition
9 }# end of govDelay dictionary

```

Figure 4.12: Governor delay dictionary definition example.

Tuples are used to enter delay block parameters. Using Figure 4.3 as reference, the ‘wDelay’ tuple contains settings for D_A , T_A , and K_A respectively. Likewise, the ‘PrefDelay’ tuple contains D_B , T_B , and K_B . If the tuple contains three entries, the third is assigned to the optional gain associated with each block, otherwise K_x is one.

4.3.2.3.7 Governor Deadband Dictionary

A BA agent may be used to set area wide deadbands, but it is also possible to specify a single deadband for any governed generator. Individual governor deadband dictionaries are defined inside the mirror.govDeadBand dictionary in the .ltd.py file. Settings in each governor deadband dictionary will override any deadband settings specified by the BA dictionary.

Figure 4.13 shows three examples of valid governor deadband definitions.

Entering ‘step’ or ‘ramp’ as a value for the ‘GovDeadbandType’ will create a step or ramp deadband at the given ‘GovDeadband’. A non-linear droop governor deadband may be configured by setting the ‘GovDeadbandType’ to ‘NLDrop’ and entering desired ‘GovAlpha’ and ‘GovBeta’ values. An explanation of the different deadband types is presented in Section

```

1 mirror.govDeadBand ={  

2     'gen3DB' : {  

3         'genBus' : 3,  

4         'genId' : None, # optional  

5         'GovDeadbandType' : 'ramp', # step, ramp, nldroop  

6         'GovDeadband' : 0.036, # Hz  

7     },  

8     'gen1DB' : {  

9         'genBus' : 1,  

10        'genId' : None, # optional  

11        'GovDeadbandType' : 'nldroop', # step, ramp, nldroop  

12        'GovAlpha' : 0.016, # Hz, used for nldroop  

13        'GovBeta' : 0.036, # Hz, used for nldroop  

14    },  

15     'gen4DB' : {  

16         'genBus' : 4,  

17         'genId' : None, # optional  

18         'GovDeadbandType' : 'step', # step, ramp, nldroop  

19         'GovDeadband' : 0.036, # Hz  

20         'GovAlpha' : 0.016, # Hz, used for nldroop  

21         'GovBeta' : 0.036, # Hz, used for nldroop  

22     },  

23 }# end of govDelay dictionary

```

Figure 4.13: Governor deadband dictionary definition example.

5.3.1.

4.3.2.3.8 Definite Time Controller Dictionary

During long simulations, system loading may change from initial values by more than $\pm 20\%$. Such changes can cause voltage issues that require the setting or un-setting of components contributing to available system reactive power. This can be accomplished by defining a definite time controller (DTC) agent in the mirror.DTCdict dictionary. Other general programmable logic operations may also be accomplished using a DTC agent. Figure 4.14 shows an example of a valid DTC definition where a shunt is actuated by changes in bus voltage or branch MVAR flow. It should be noted that instead of using a specific ‘tarX’ in a timers ‘act’ field, operations on any off or on target can be accomplished by using

‘anyOFFtar’ or ‘anyONTar’ respectively.

```

1  # Definite Time Controller Definitions
2  mirror.DTCdict = {
3      'ExampleDTC' : {
4          'RefAgents' : {
5              'ra1' : 'bus 8 : Vm',
6              'ra2' : 'branch 8 9 1 : Qbr', # branches defined from, to, ckID
7          },# end Reference Agents
8          'TarAgents' : {
9              'tar1' : 'shunt 8 2 : St',
10             'tar2' : 'shunt 8 3 : St',
11         }, # end Target Agents
12         'Timers' : {
13             'set' :{
14                 'logic' : "(ra1 < 1.0) or (ra2 < -15)",
15                 'actTime' : 30, # seconds of true logic before act
16                 'act' : "tar1 = 1",
17             },# end set
18             'reset' :{
19                 'logic' : "(ra1 > 1.04) or (ra2 > 15)",
20                 'actTime' : 30, # seconds of true logic before act
21                 'act' : "tar1 = 0",
22             },# end reset
23             'hold' : 60, # minimum time between actions
24         }, # end timers
25     },# end ExampleDTC definition
26 }# end DTCdict

```

Figure 4.14: Definite time controller dictionary definition example.

Each DTC employs a set and a reset timer and may have a hold timer if hold time is set larger than zero. Multiple references and targets can be associated with a DTC, however, as of this writing only one action can be associated with each timer. Any logic string entered in a timer uses the given key names for each reference or target and is evaluated using standard Python logic conventions.

4.3.3 Simulation Initialization

To clarify the explanation of simulation initialization, the entire process can be broken into three parts: process creation, mirror initialization, and dynamic initialization pre-simulation loop. The first part (process creation) involves creating and configuring various software processes that enable the simulation to run. The majority of part two (mirror initialization) revolves around collecting data from PSLF to create a Python duplicate, or mirror, of the power system model. The last part of simulation initialization (dynamic initialization pre-simulation loop) handles user input and creates PY3 dynamics before entering the simulation loop.

4.3.3.1 Process Creation

System initialization begins in PY3 with package imports and creation of truly global variables before user input from the .py file is handled. The .py file includes debug flags, simulation notes, simulation parameters, and file locations of the .sav, .dyd, and .ltd.py files. If all file locations are valid, PY3 initializes AMQP queues, sends appropriate initialization information to the IPY queue, starts the IPY_PSLTDSim process, and then waits for an IPY response message.

The IPY process begins by also importing required packages and setting references to certain imported packages as truly global variables. An IPY AMQP agent is created and linked to the AMQP host generated by the PY3 process. This allows the IPY process to receive initialization information from the PY3 AMQP message sent before the IPY process was evoked. IPY uses the received initialization information to load the GE Python API which is then used to load the .sav and .dyd into PSLF. Upon successful file loading in PSLF, a global reference to the object is created.

4.3.3.2 Mirror Initialization

Once the PSLF specific files are loaded into the GE software, initialization of the Python environment, or system model, may begin. The system model, referred to as the system mirror, or just mirror, is a single object that almost all other Python objects are

created inside. The mirror is a single system object with a recursive data structure that allows any object the ability to reference any other object as long as they both share a reference to the mirror and are themselves referenced by the mirror. While the previous sentence may seem overly complicated, its not, and the use of such a linking technique eliminated the need for global variables outside of imported packages and also allowed for a single file containing **all** simulation data to be easily exported at the end of a simulation.

The system mirror begins its initialization by creating placeholder variables for simulation parameters, counters, and future agent collections. Specific case parameters, such as the number of buses or generators, are collected from PSLF to be used later in model verification processes.

Before any specific power system object data is collected, an initial power flow is performed to ensure that the loaded .sav is solvable, and to establish a steady state system operating point. System agents, representing power system objects like buses and loads, are then added to the mirror. The adding process queries PSLF for any buses in a specific area, checks each found bus for any connected system components, and creates Python agents for relevant objects. The querying and adding process continues until all system buses are accounted for. Each type of found object is added to a running tally so that it can be compared with the expected values collected earlier. Once all system buses have been found or accounted for, any created agents that are intended to log data are collected into a list for simpler group stepping.

Each area tally of found agents is checked for coherency with expected values. Any inconsistencies between the amount of found and expected objects will trigger warnings, but the simulation will not stop if this occurs. This choice is due to differences between what is counted in PSLF as a valid area object and PSLTDSim, which has the option to ignore islanded objects.

Dynamic model information from the specified .dyd file is then parsed. Collected machine or governor parameters are used to create PSLF model information objects (PMIOs)

inside the mirror. These PMIOs collect inertia H and MW cap for each machine as well as turbine type, governor MW cap, and permanent droop R from governors. Other information, such as MVA base, is also collected for both types of model. This process is required as the .dyd values for certain parameters overwrite pre-existing values that may be saved inside the .sav file.

Once the .dyd file is parsed and all PMIOs are created, the combined system inertia is calculated. For each found generator PMIO, the associated mirror agent is located and updated so that H and Mbase values match those found in the .dyd. The total system inertia is calculated according to Equation 4.1 and user input settings are interpreted so that any requested changes, such as scaling or alternative system inertia inputs, are handled correctly.

Mirror search dictionaries are then created to simplify and speed up agent searches and the global slack generator is identified. The global slack generator is important to locate as its variance from an expected value dictates accelerating power re-distribution and power-flow solution iterations during the simulation loop. Once search dictionaries are created, the IPY model is fully initialized. The mirror is then saved to disk and an AMQP message is sent to PY3 with the mirror location.

4.3.3.3 Dynamic Initialization Pre-Simulation Loop

After the handoff AMQP message is sent to PY3, IPY initializes values required for simulation and awaits an AMQP message from PY3 before entering its simulation loop.

PY3 uses the information received from IPY to load the newly created system mirror so that PY3 can perform further initializations such as creating any dynamic agents (governors), calculating area frequency characteristics and maximum capacities, executing any .ltd code, and creating the associated agents from the .ltd input.

PY3 dynamics are initialized to ensure R is on the correct PU base and any MW caps from dyd parsing is applied. Additionally, any limiting values for governor output are accounted for, and any deadbands or delays are created. Finally, before entering the PY3 simulation loop, agents that are designed to log values initialize blank lists for expected

values.

4.3.4 Simulation Loop

Once simulation initialization is complete, and the system mirror is initialized, the simulation loop is executed. Figure 4.15 shows the major actions that are processed each time step, but does not include details concerning AMQP communication. AMQP messages are sent and received during the ‘Update’ blocks and the system mirrors are checked for coherency at these times as well.

The simulation loop can be viewed as starting with the increment of simulation time followed by the stepping of any dynamic agents. These dynamic agents calculate the new system frequency and perform any required governor responses. Generator electrical power is then set equal to generator mechanical power. This step is the beginning of forming the next power-flow solution initial conditions.

Perturbation agents are then stepped. This means that any steps, ramps, or noise type events are performed, agents in both system mirrors are updated, and any related system value is changed accordingly. For example, the tripping of a generator requires the system inertia to change as well as the amount of power in the system. The variable ΔP_{pert} is used to keep track of system power changes. Additionally, BA action takes place at this time.

After all perturbation related actions are executed, accelerating power is calculated and distributed to the system according to generator inertia. The PSLF system is updated with any required values and a power-flow solution is attempted. If the solution diverges the simulation ends and any collected data is output. If the solution does not diverge, the magnitude of any slack error is checked against the slack error tolerance. If the slack error is larger than the slack tolerance, the error is redistributed to the system until the resulting error is within tolerance or the solution diverges. If the power-flow solution diverges during accelerating power redistribution the simulation ends and any collected data is output.

Once the system has converged to a point where the slack error is less than the slack tolerance, PSLF values for generator real and reactive power and bus voltage and angle are used to update the PY3 mirror. The electric power output of the system is summed for use

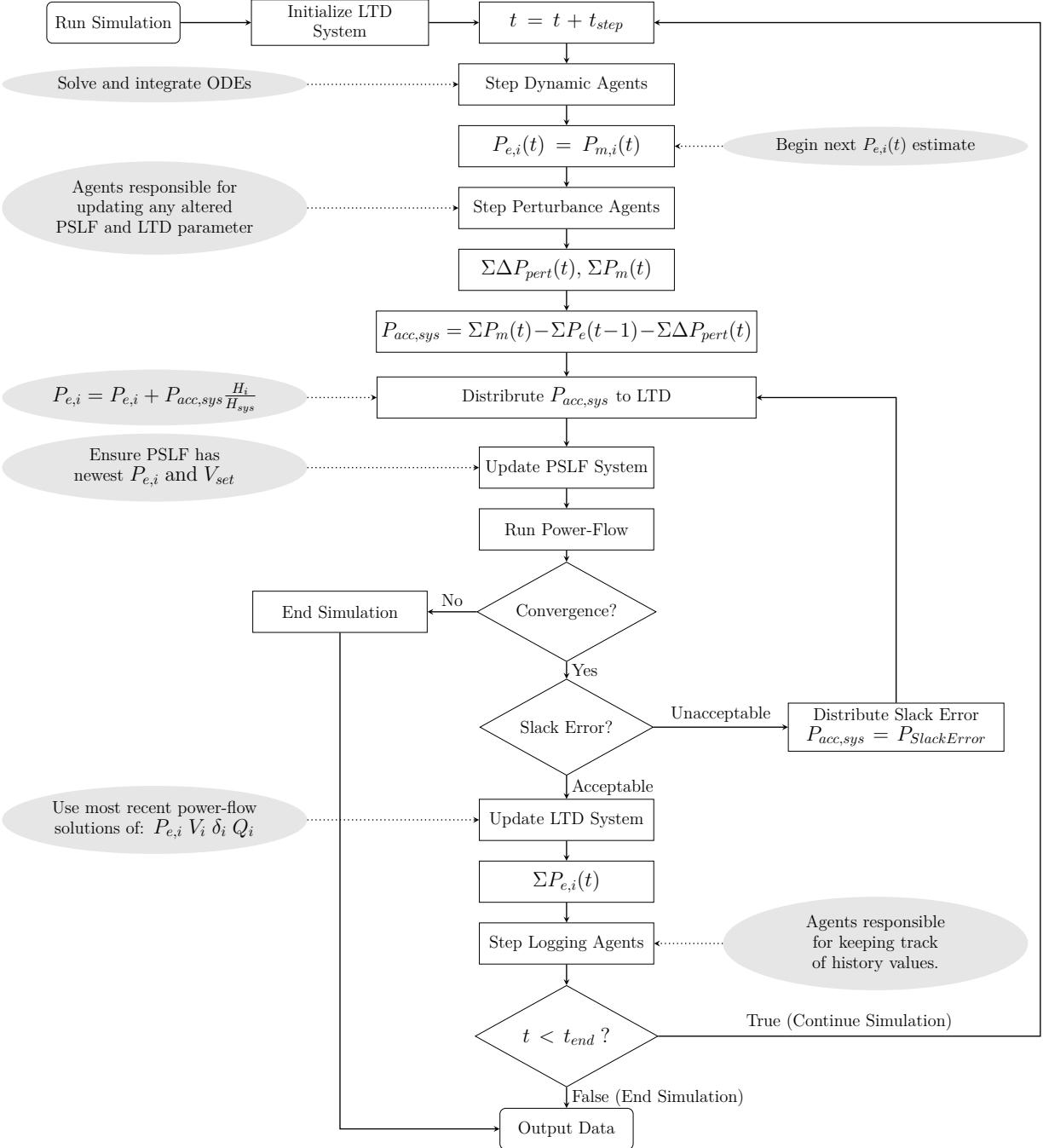


Figure 4.15: Simulation time step flowchart.

in calculating system accelerating power in the next time step. Any logging agents are then stepped and data for that particular simulation time step are recorded. Finally, system time is checked and if the simulation is complete, any collected data is output. If the simulation is not complete, the simulation time is incremented by the time step and the cycle repeats

itself again.

4.3.5 Simulation Outputs

Pre-defined data is collected by any agent with logging ability. Current agents with this ability are machines, loads, shunts, branches, transformers, areas, balancing authorities, and the system mirror itself. When a simulation is complete, the final system mirror is exported via the Python package `shelve` and options exist to export some data as a MATLAB .mat file. The .mat output is accomplished by combining various agent log dictionaries into a single dictionary. As such, only data deemed useful for validating the software is included.

It should be noted that numerous plot functions were created to easier visualize and validate python mirror data. Python plot functions are located in the PSLTDSim package plot folder, while the MATLAB validation plots are located in the GitHub repository only.

4.4 Software Validation

PSLTDSim was validated by comparing results from identical simulation scenarios performed in PSDS and PSLTDSim. For clarification, data from PSLTDSim, or simulations performed in PSLTDSim, are described as LTD data or simulations, respectively. Compared values of interest include: system frequency, generator mechanical power, generator real power, bus voltage magnitude, bus voltage angle, generator reactive power, branch current, branch real power flow, and branch reactive power flow. The following sections describe the plots used for validation, present results from various scenarios, and provide a validation summary.

4.4.1 Validation Plots Explained

Plots were used to present simulation validation data. Due to the volume of information, various types of plots were created. Figure 4.16 shows the three types of plots used for the validation process. Because of the different time steps used in PSDS and LTD, when computing difference data between the two simulations, the same LTD value was held for multiple comparisons. For example, any PSDS($t = 2.x$) data was compared to the corresponding LTD($t = 2$) data point.

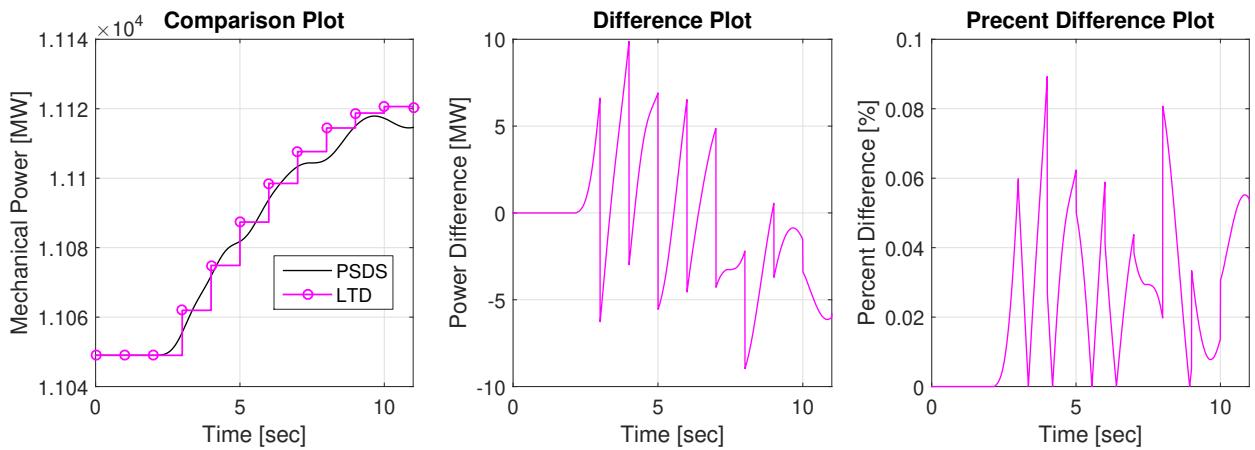


Figure 4.16: Validation plot examples.

4.4.1.1 Comparison Plot

The most basic plot is a comparison plot that simply puts data from one simulation environment on top of another. Two examples of a comparison plot is shown in Figure 4.17 where LTD data is shown using open circles and PSDS data plotted as filled dots.

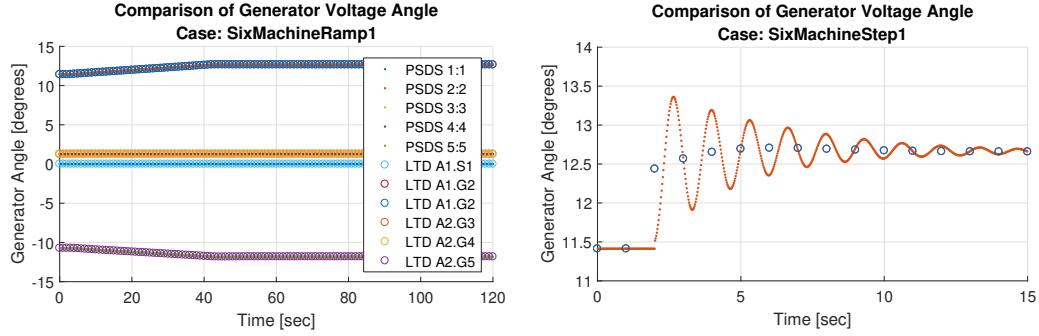


Figure 4.17: Comparison plot examples.

While comparison plots are useful for quick validation, they do not provide substantive quantitative data. Additionally, when comparing many values, plots become difficult to interpret. As a result, comparison plots are typically only used for showing frequency.

4.4.1.2 Difference Plot

To allow for more easily compared values and provide quantitative data, a difference plot was created. Figure 4.18 provides an example of a difference plot. Since individual comparisons seemed less important, all data comparisons were plotted in grey with an absolute average plotted in black.

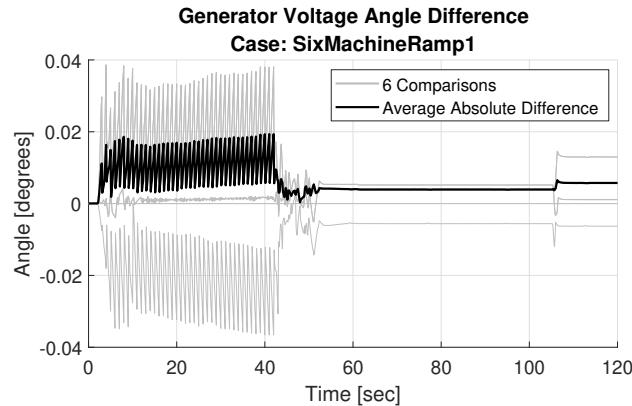


Figure 4.18: Difference plot example.

Equation 4.6 describes the difference calculation between PSDS and LTD variables.

$$\text{Difference}_{data} = \text{PSDS}_{data} - \text{LTD}_{data} \quad (4.6)$$

The absolute average was calculated using Equation 4.7 where $data_i$ represents a time series of difference data for a particular value of interest and n is the total number of comparisons made.

$$\text{Average}_{abs} = \frac{\sum_i^n |data_i|}{n} \quad (4.7)$$

While difference plots are more quantitative than comparison plots, there is no sense of scale when comparing two values. As power systems can greatly vary in size, a subjectively insignificant difference in one system may be very significant in another system.

4.4.1.3 Percent Difference Plot

To account for the wide variety of data magnitudes being compared, a percent difference plot was created. Figure 4.19 shows two examples of a percent difference plot.

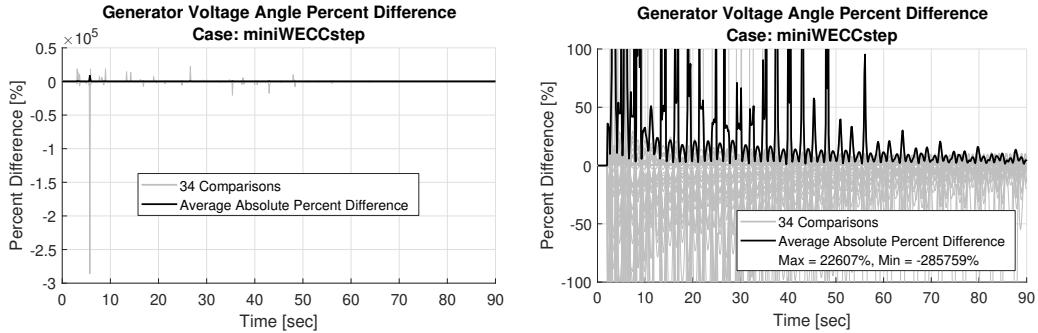


Figure 4.19: Percent difference plot examples.

Similar to the difference plot, individual comparisons were not deemed as important a cumulative comparison, so the same plotting scheme is repeated. The percent difference was calculated according to Equation 4.8.

$$\%_{diff} = \frac{|\text{PSDS}_{data} - \text{LTD}_{data}|}{\frac{\text{PSDS}_{data} + \text{LTD}_{data}}{2}} \times 100\% \quad (4.8)$$

When using percent difference, it should be noted that if one value is positive, and the other negative, results may be misleading. More specifically, as the average of the two numbers being compared approaches zero, so does the denominator of Equation 4.8. Such a situation may lead to a divide by zero error or, more likely, a very large percent difference. To alleviate such data obfuscation, the y-axis was scaled if the average absolute percent difference was over 150% and the maximum and minimum percent differences are listed in the legend. This is shown in the right hand plot of Figure 4.19.

4.4.1.4 Weighted Frequency Plot

Frequency comparisons were used to validate the single system frequency assumption calculated by PSLTDSim. Comparison of frequency data from PSDS to LTD was simplified by calculating a single weighted PSDS frequency f_w based on generator inertia. In a system with N generators

$$f_w = \sum_{i=1}^N f_i \frac{H_{PU,i} M_{base,i}}{H_{sys}} \quad (4.9)$$

where f_i is a full time series of machine i 's frequency, and H_{sys} is calculated according to Equation 4.1.

Figure 4.20 shows all bus frequencies plotted in grey, the calculated weighted frequency in black, and the single system frequency calculated by PSLTDSim in magenta. It is worth pointing out that non-generator bus frequencies are ignored in the weighted system frequency (because they lack inertia), and that the weighted frequency was used for difference calculations between PSDS and PSLTDSim.

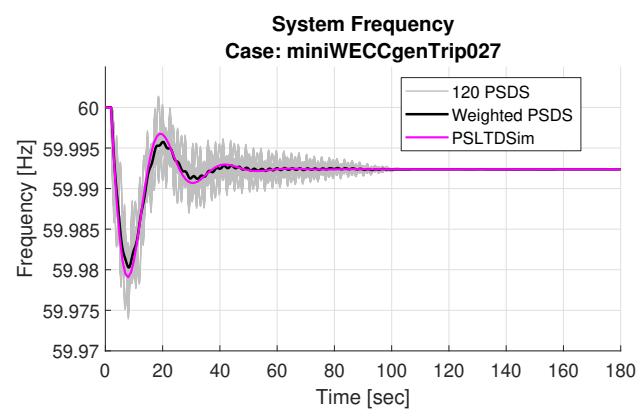


Figure 4.20: Frequency comparison plot example.

4.4.2 Six Machine System

A two area six machine system used for validation is shown in Figure 4.21. The system is based off the Kundur four machine system presented in [35] with some modifications.

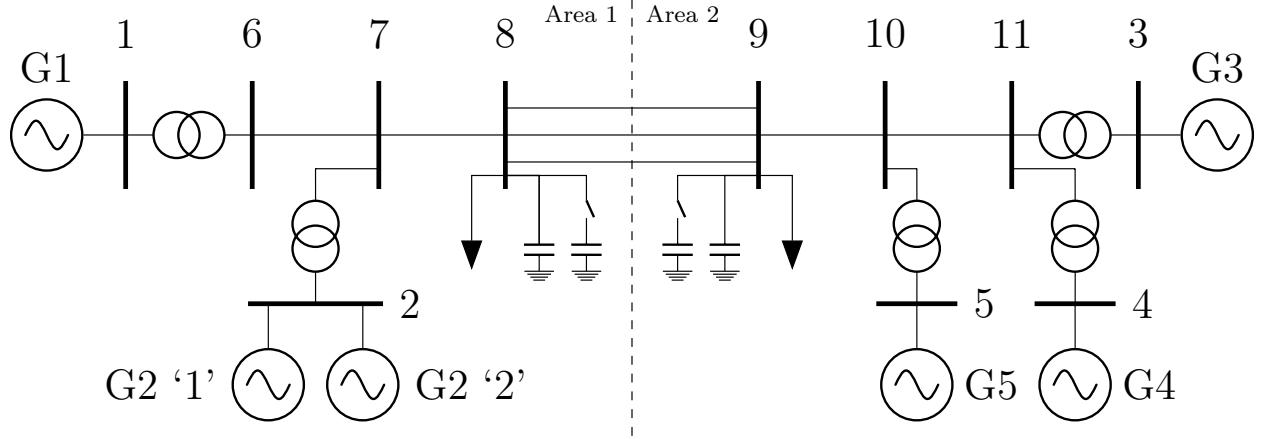


Figure 4.21: Six machine system.

The most noticeable difference is the addition of two generators and switchable shunts on bus 8 and 9. The generators were added to test power plant style AGC dispatching and multiple generators per bus. While placing multiple generators per bus is not a standard or recommended modeling technique, it is present in the full WECC PSLF model and as such, should be validated in PSLTDSim. Detailed six machine system model specifications are presented in Appendix A.

4.4.2.1 Simulated Scenario Descriptions

A load step, a load ramp, and a generator trip were simulated using the six machine model. The simplest validation test, a stepping of load, occurred at $t = 2$ when the load on bus 9 was increased by 75 MW. To test longer perturbances, a 40 second 75 MW load ramp of the load on bus 9 was simulated. Due to the LTD one second time step, the ramp is equivalent to forty 1.875 MW steps taking place every second from $t = 2$ to $t = 42$. Finally, to test the handling of inertia and a slightly larger step perturbation, generator 5, which was initially generating 90 MW, was tripped at $t = 2$.

4.4.2.2 Frequency Results

System frequency comparison results are presented for each simulated scenario in Figures 4.22, 4.23, and 4.24. The load step results show a maximum 18 mHz difference that is reduced below 2.5 mHz by $t = 25$. The system frequency difference during the ramp test never exceeds more than 1.4 mHz with peaks at the start and end of the ramp event. Similar to the load step result, the generator trip event had a maximum difference of 30 mHz that was reduced below 2.5 mHz by $t = 25$ and essentially matched the PSDS frequency. The larger frequency peak difference is due to the generator trip perturbation being 15 MW larger than the load step perturbation.

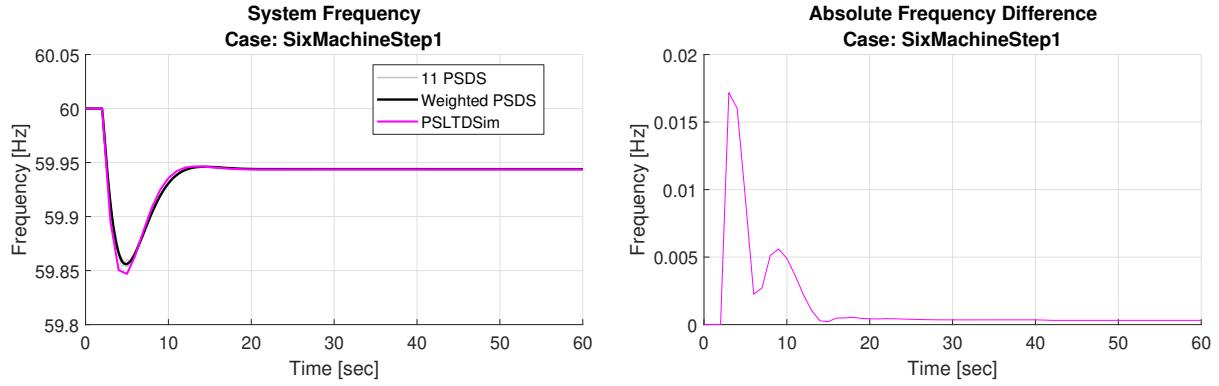


Figure 4.22: Six machine load step system frequency comparison.

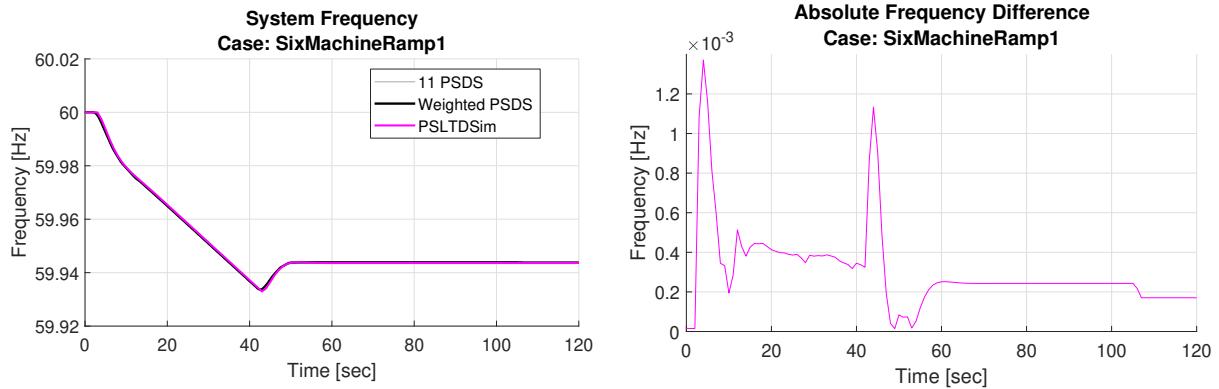


Figure 4.23: Six machine load ramp system frequency comparison.

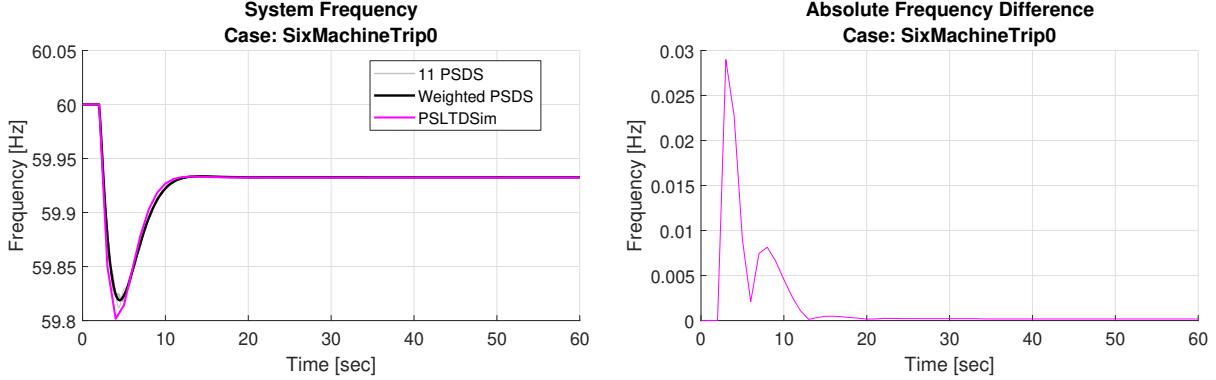


Figure 4.24: Six machine generator trip system frequency comparison.

4.4.2.3 Generator Mechanical Power Results

Generator mechanical power comparison results are presented for each simulated scenario in Figures 4.25, 4.26, and 4.27. It should be noted that only generators equipped with governors are compared in this section as machines without governors are not modeled to have any changes in mechanical power. Similar to frequency results, the ramp event had the smallest difference from PSDS never exceeding 0.5 MW, or 0.2% difference. Unlike the frequency results, most ramp mechanical power differences happen during the ramp with no large peaks at the beginning or end of the event. Further, there is a small steady state error present after the ramp is complete. Both step type events have an immediate peak difference of 5 to 7 MW (2.5 to 3.5 percent difference) that is reduced to roughly zero after 20 seconds.

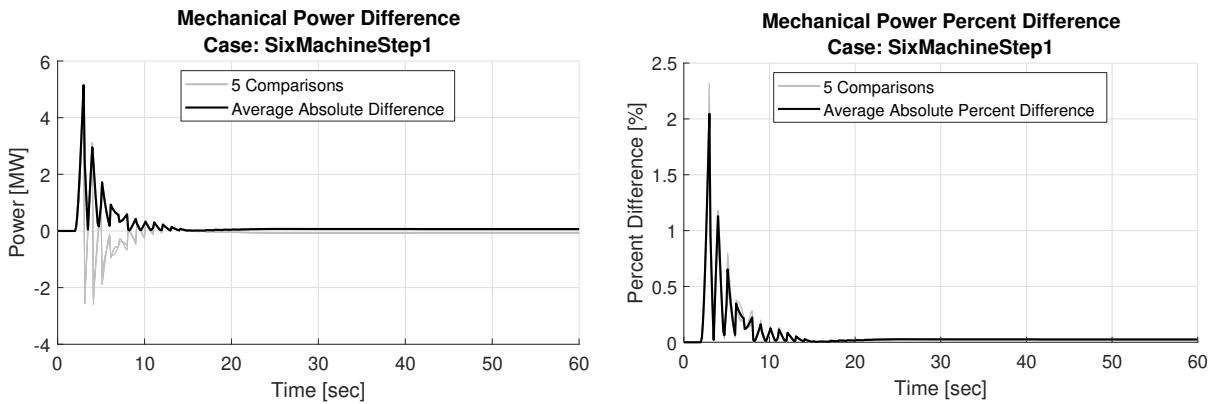


Figure 4.25: Six machine load step mechanical power comparison.

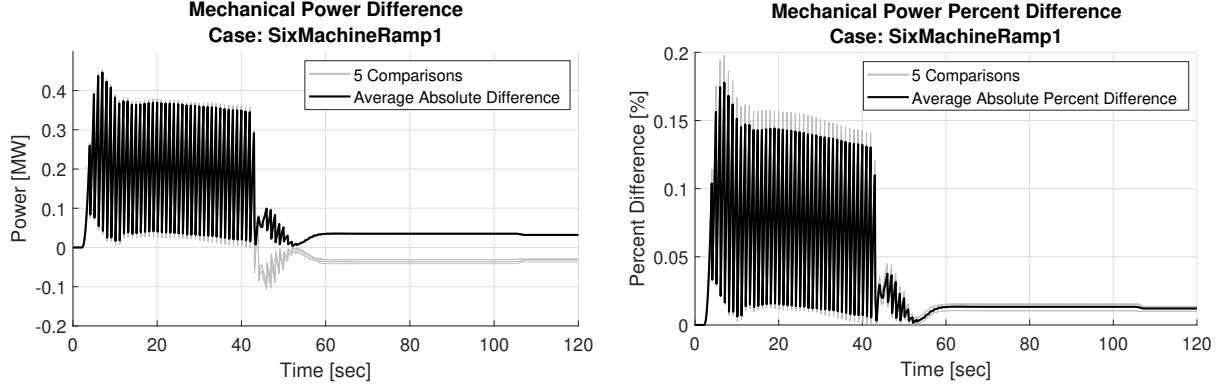


Figure 4.26: Six machine load ramp mechanical power comparison.

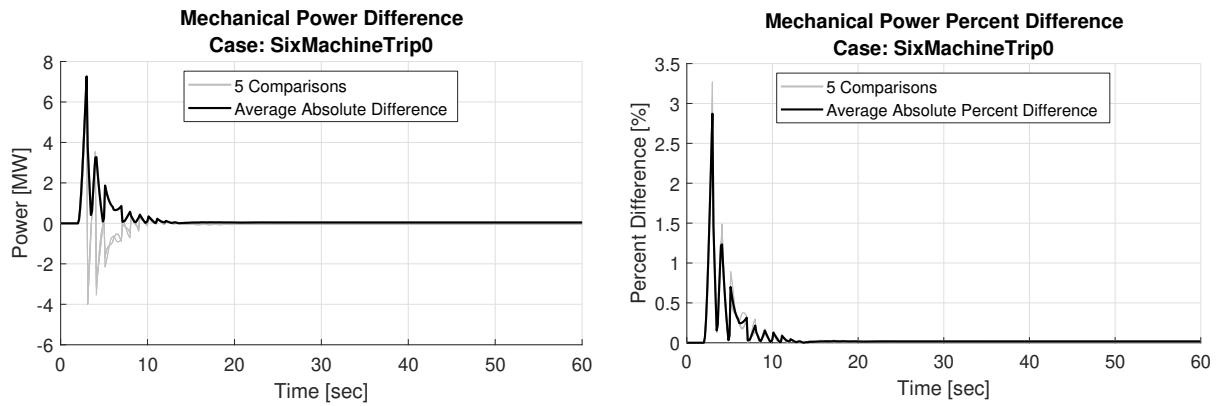


Figure 4.27: Six machine generator trip mechanical power comparison.

4.4.2.4 Generator Real Power Results

Generator real power comparison results are presented for each simulated scenario in Figures 4.28, 4.29, and 4.30. All six generators are compared in this section as real power is calculated via the power-flow solution and can change every step. Averaged results are very similar to the mechanical power results though slightly more transient. In the step type events, individual difference peaks are nearly double the mechanical power peak differences. In all cases, the difference approaches zero approximately 20 seconds after each perturbation is complete. The PSDS simulation shows a strange peak of activity during the ramp event near $t = 110$.

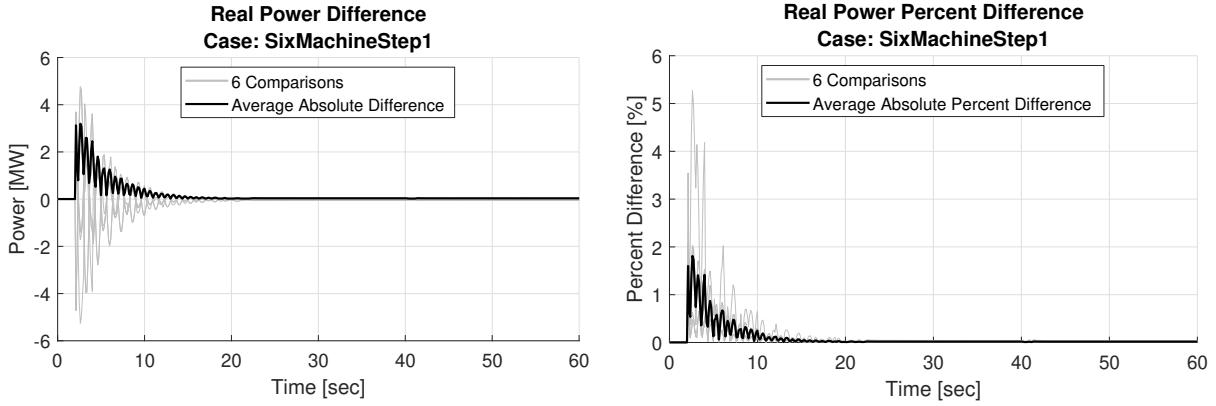


Figure 4.28: Six machine load step real power comparison.

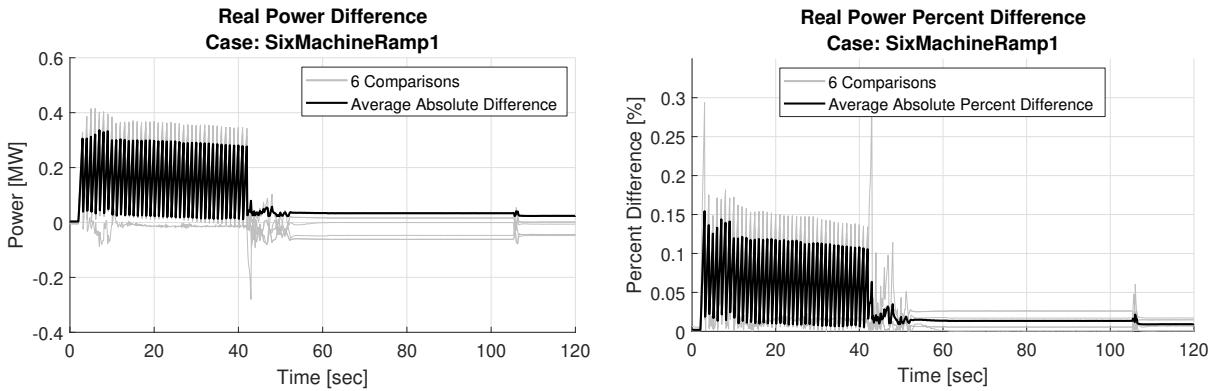


Figure 4.29: Six machine load ramp real power comparison.

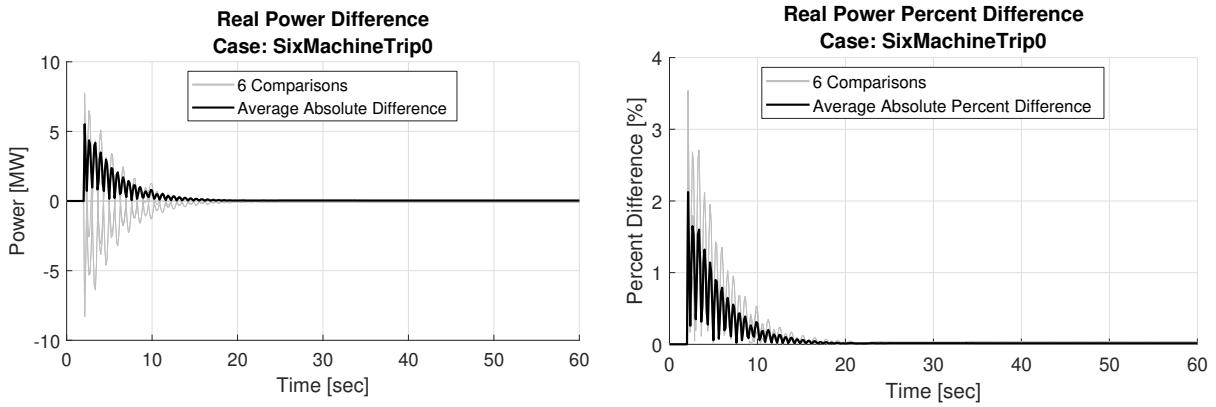


Figure 4.30: Six machine generator trip real power comparison.

4.4.2.5 Voltage Magnitude Results

Bus voltage magnitude comparison results are presented for each simulated scenario in Figures 4.31, 4.32, and 4.33. Since voltage is presented in PU, percent difference plots

appear as a scaled absolute version of the difference plots. The largest bus voltage difference, of approximately 1.5%, occurred during the generator trip test. As with the load step test, simulation results converge after $t = 20$. Voltages gradually drift during the ramp event, but stop once the event is over. The gradual voltage change is believed to be due to the difference in exciter modeling between PSDS and PSLTDSim.

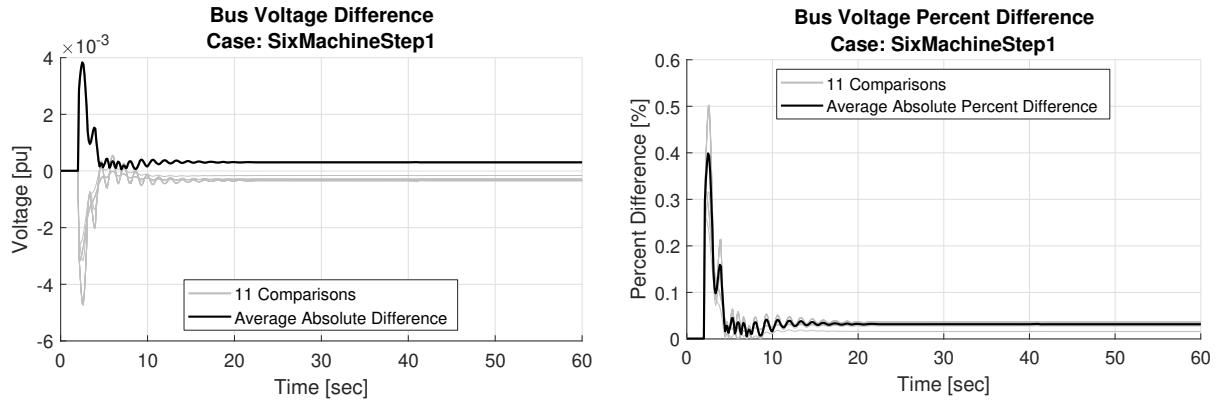


Figure 4.31: Six machine load step voltage comparison.

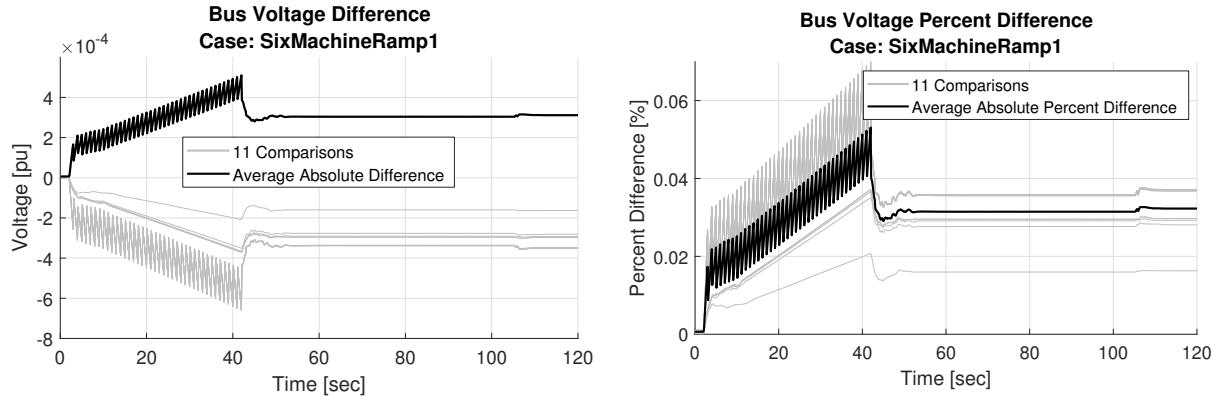


Figure 4.32: Six machine load ramp voltage comparison.

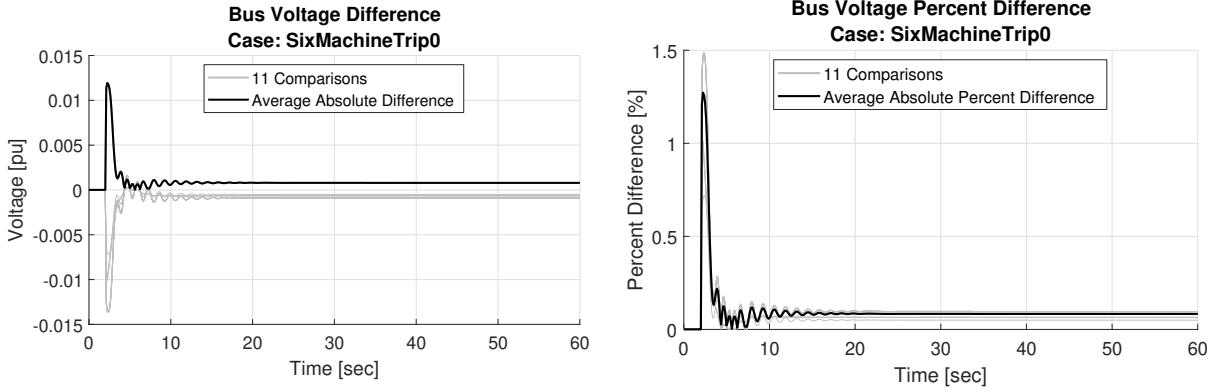


Figure 4.33: Six machine generator trip voltage comparison.

4.4.2.6 Voltage Angle Results

Bus voltage angle comparison results are presented for each simulated scenario in Figures 4.34, 4.35, and 4.36. The step type perturbances caused larger oscillatory differences than the load ramp event, but steady state angle differences converge near zero. Maximum percent difference during the generator trip reached nearly 20% while the load step and ramp had maximums of approximately 7.0% and 0.6% respectively.

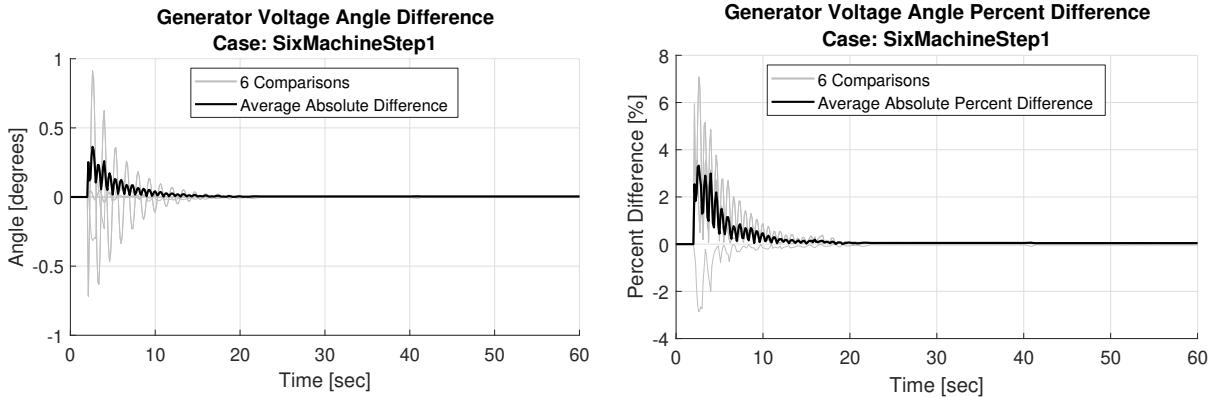


Figure 4.34: Six machine load step voltage angle comparison.

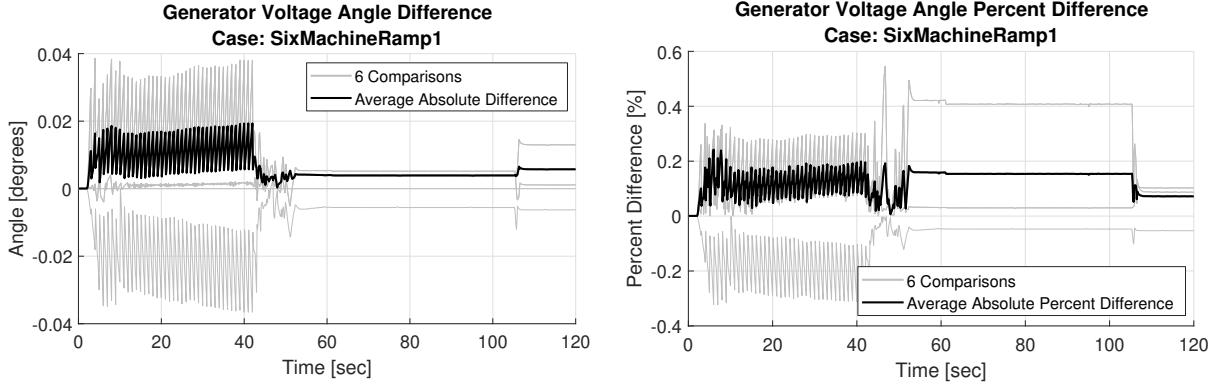


Figure 4.35: Six machine load ramp voltage angle comparison.

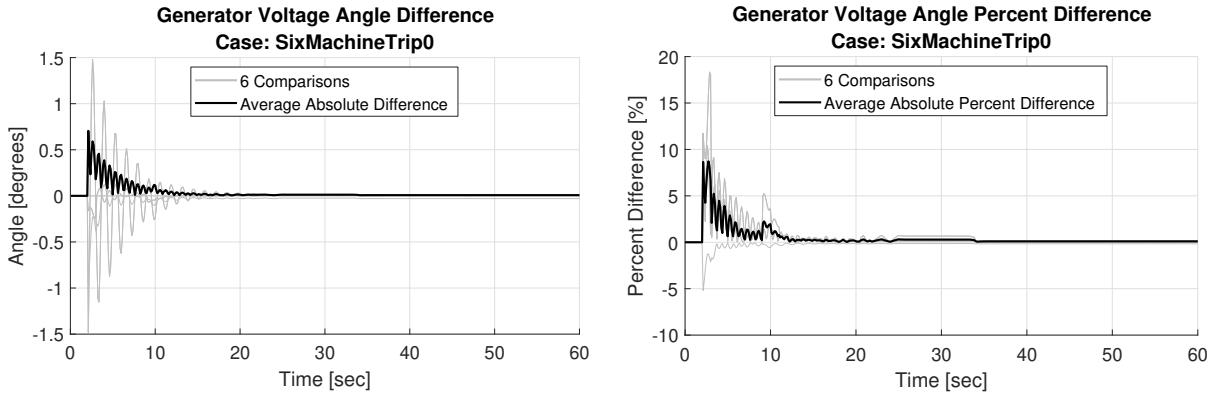


Figure 4.36: Six machine generator trip voltage angle comparison.

4.4.2.7 Generator Reactive Power Results

Generator reactive power comparison results are presented for each simulated scenario in Figures 4.37, 4.38, and 4.39. Maximum reactive power percent differences for the generator trip, load step, and ramp events were 8.0%, 5.0%, and 0.8% respectively. As in previous comparisons, the step events had large oscillatory differences immediately following the event, but have very low steady state error by $t = 20$. The ramp event had less than 1.0% differences during the event and less than 0.2% difference in the steady state.

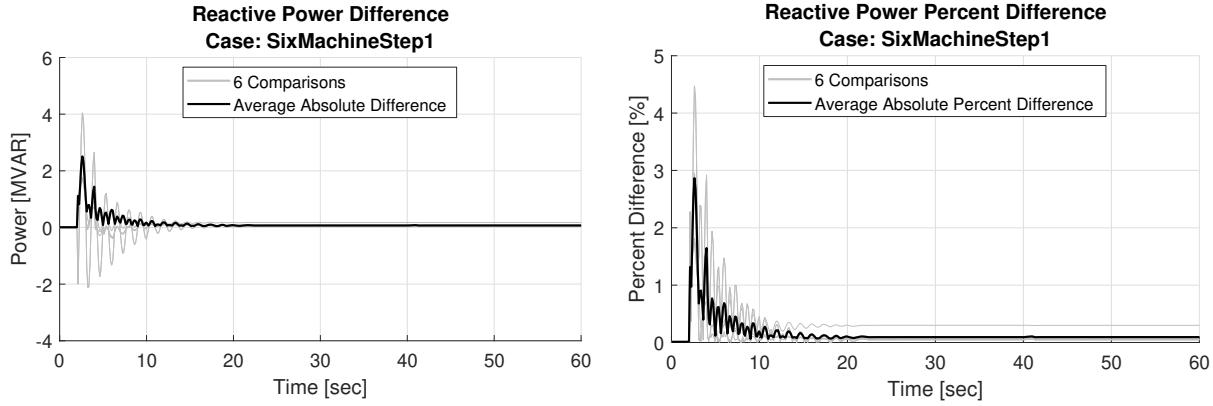


Figure 4.37: Six machine load step reactive power comparison.

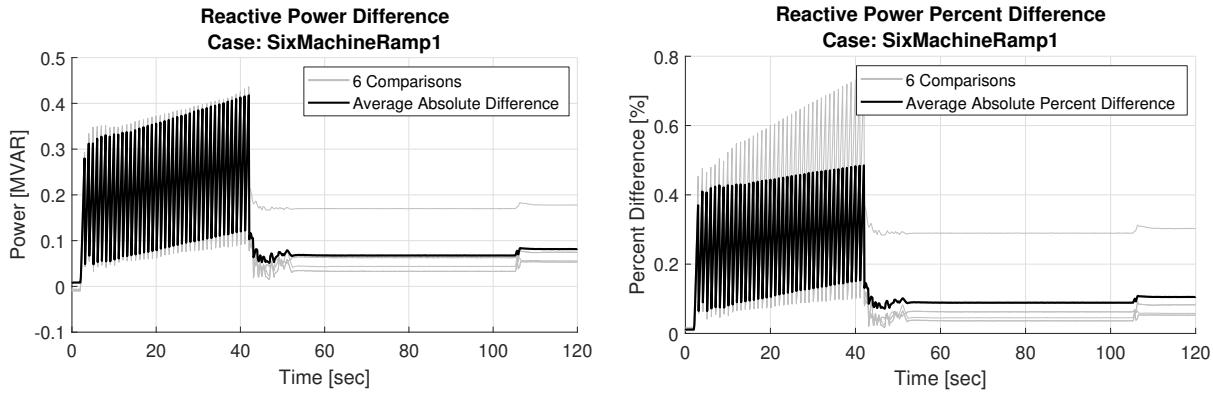


Figure 4.38: Six machine load ramp reactive power comparison.

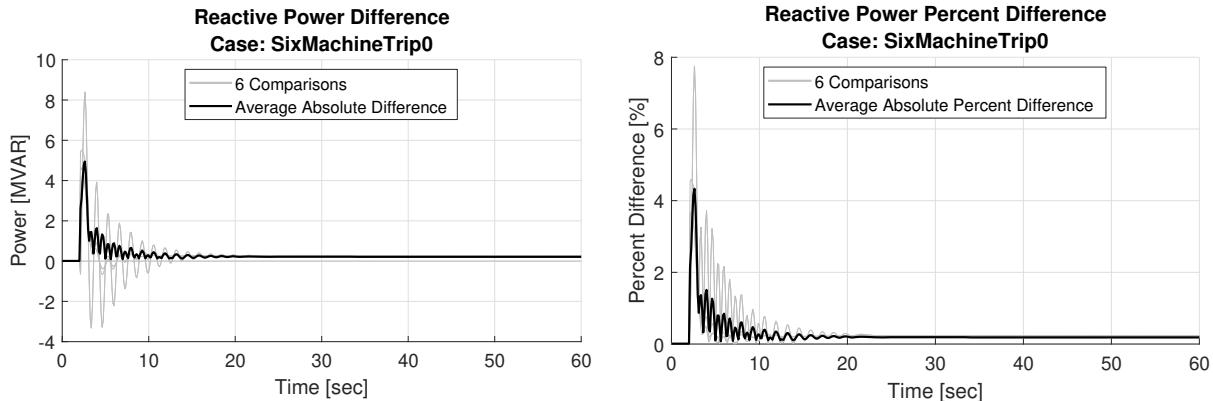


Figure 4.39: Six machine generator trip reactive power comparison.

4.4.2.8 Branch Current Results

Branch current comparison results are presented for each simulated scenario in Figures 4.40, 4.41, and 4.42. Maximum branch current percent differences for the generator trip, load

step, and ramp events were 8.5%, 7.0%, and 1.2% respectively. Again, largest differences occur immediately following the event while steady state differences are near zero.

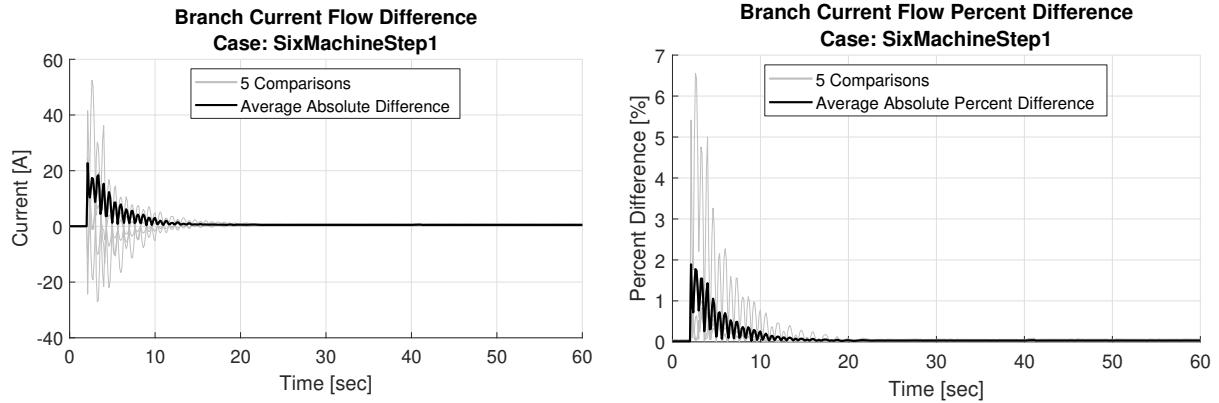


Figure 4.40: Six machine load step branch current flow comparison.

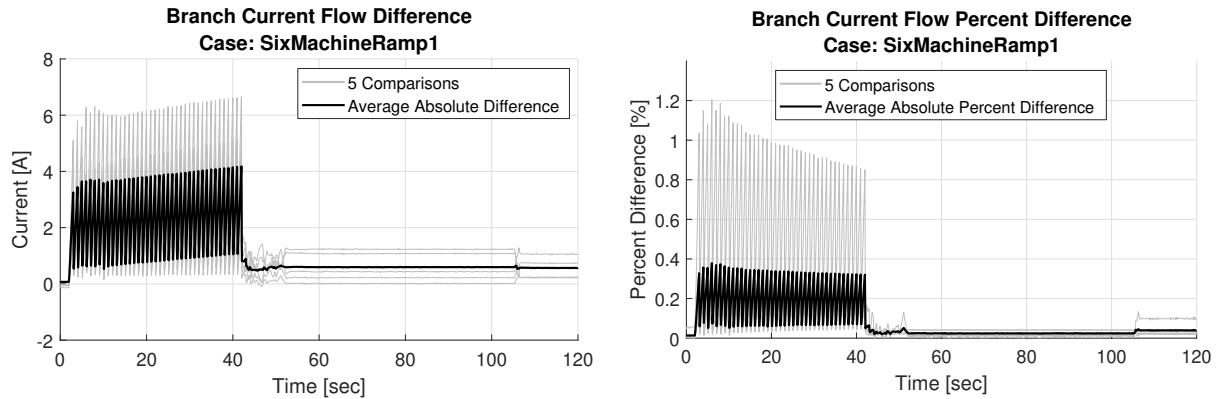


Figure 4.41: Six machine load ramp branch current flow comparison.

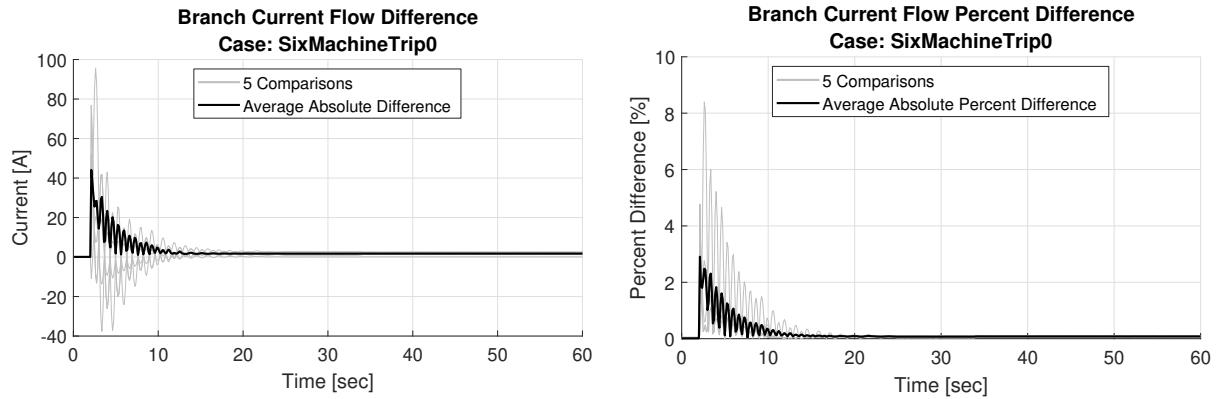


Figure 4.42: Six machine generator trip branch current flow comparison.

4.4.2.9 Branch Real Power Flow Results

Branch real power flow comparison results are presented for each simulated scenario in Figures 4.43, 4.44, and 4.45. With results very similar to branch current flow, branch real power flow percent difference maximums for the generator trip, load step, and ramp events were 8.5%, 7.0%, and 1.2% respectively. Differences in the steady state are near zero.

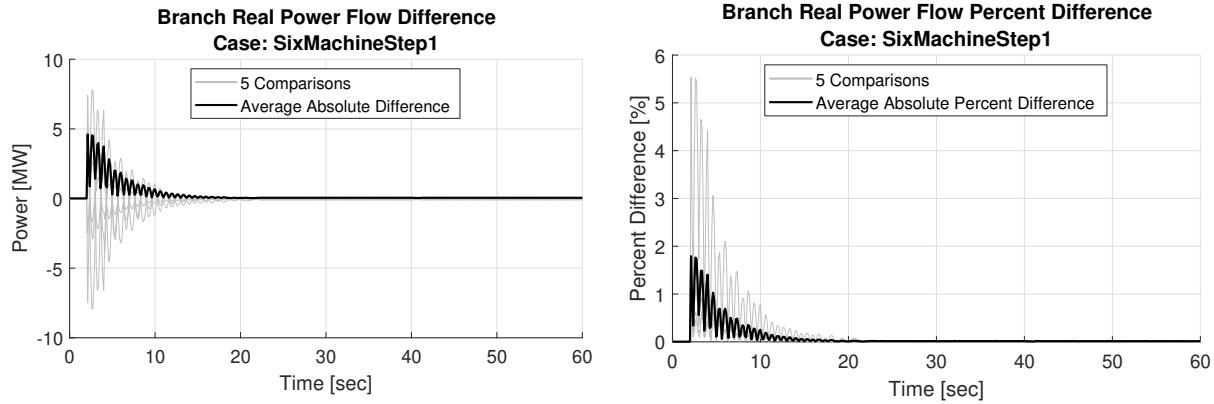


Figure 4.43: Six machine load step branch real power flow comparison.

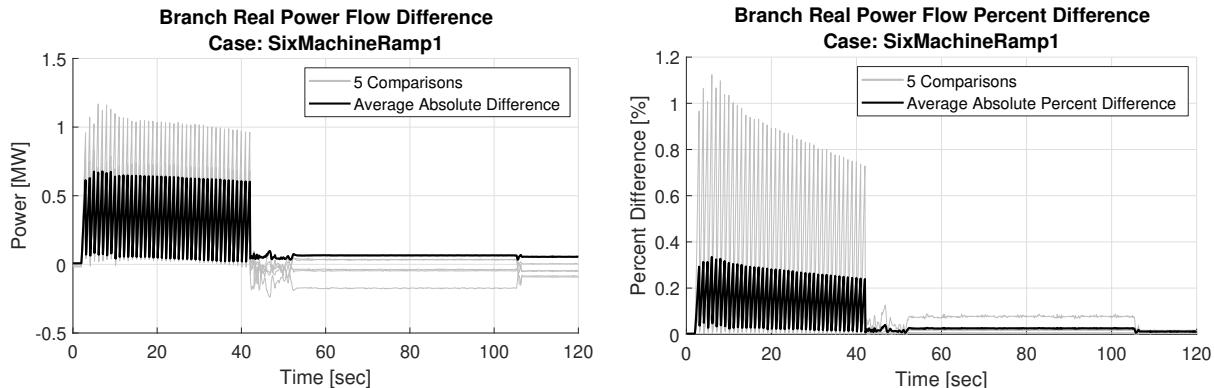


Figure 4.44: Six machine load ramp branch real power flow comparison.

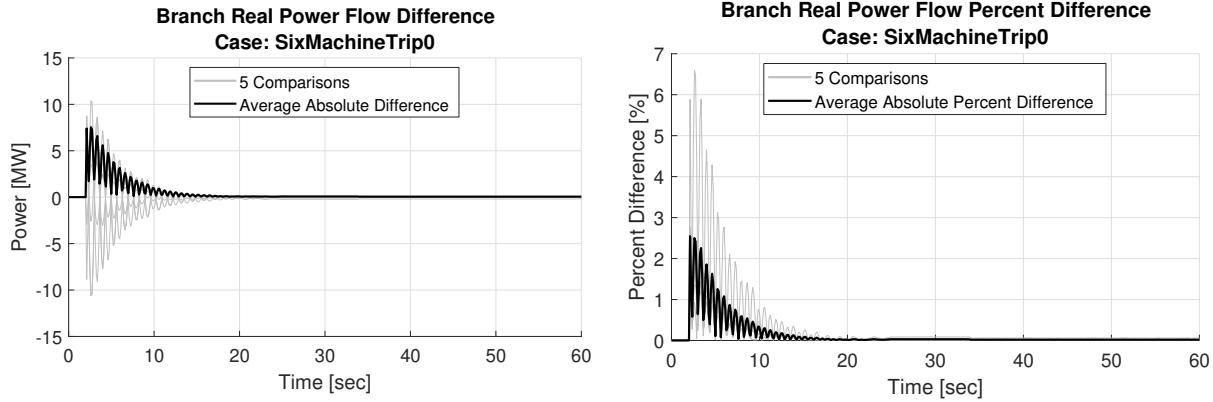


Figure 4.45: Six machine generator trip branch real power flow comparison.

4.4.2.10 Branch Reactive Power Flow Results

Branch reactive power flow comparison results are presented for each simulated scenario in Figures 4.46, 4.47, and 4.48. While the timing of value differences in branch reactive power flow is similar to current and real power flow, the magnitude of difference is not. Maximum branch reactive power flow percent differences for the generator trip, load step, and ramp events were 45.0%, 45.0%, and 5.0% respectively. These large differences may be due to the size of the system and actual MVAR flow on lines being low. Despite the large peak differences, steady state values from step events approaches zero while the ramp event had steady state differences of nearly $\pm 2.0\%$.

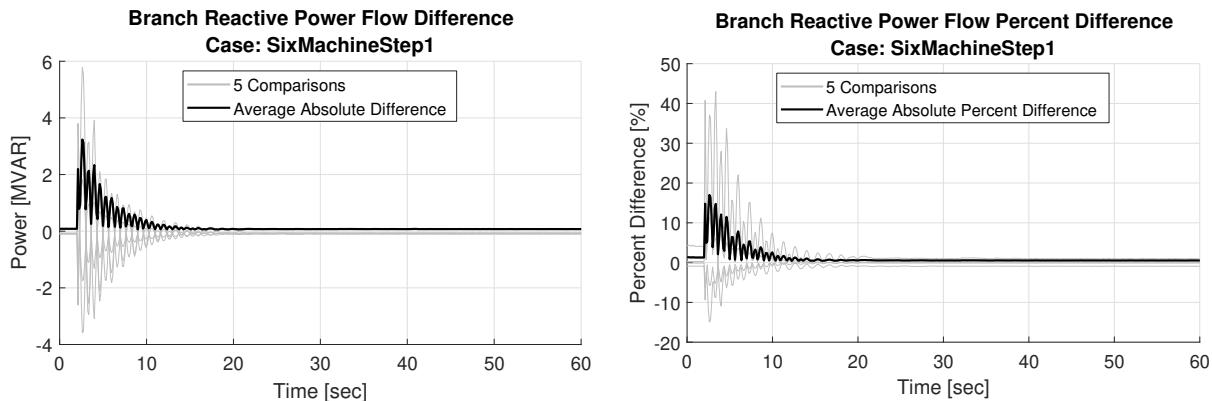


Figure 4.46: Six machine load step branch reactive power flow comparison.

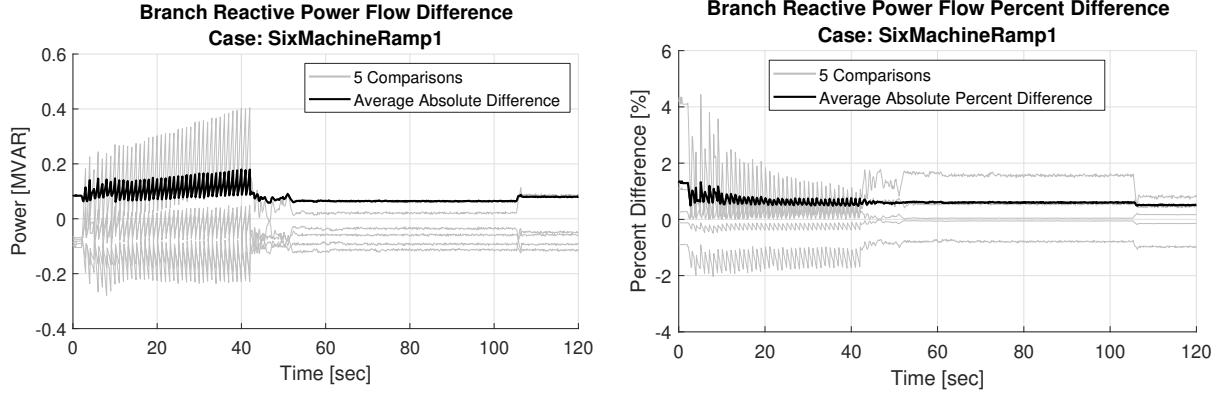


Figure 4.47: Six machine load ramp branch reactive power flow comparison.

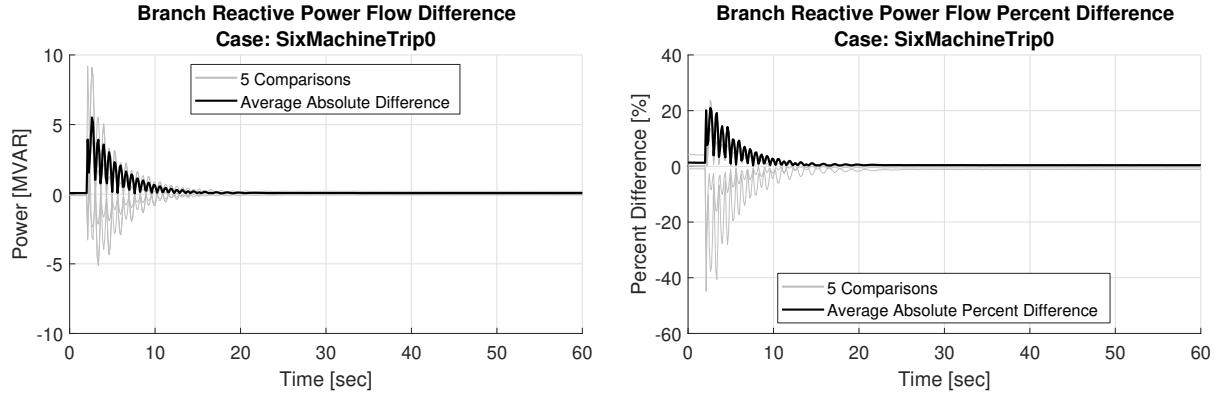


Figure 4.48: Six machine generator trip branch reactive power flow comparison.

4.4.2.11 Six Machine Result Summary

Nearly all percent differences were less than 10.0% and approached zero in the long-term. Calculated system frequency never exceeded a ± 3.0 mHz difference from the PSDS weighted frequency. Smaller perturbances, such as the ramp event, match PSDS results better than the large step event of a generator trip. Reactive power flow differences for step type events had the largest peak differences from PSDS simulation. Minor variations in voltage magnitude and angle caused by ideal exciter assumptions are believed to be the main contributer to reactive power flow difference.

4.4.3 Mini WECC System

A larger system model was used to test the scaling ability of PSLTDSim . The ‘mini WECC’, shown in Figure 4.49, is a 120 bus 34 generator system model created in PSLF. Total generation of 102,050 MW supplies 100,685 MW of load over 104 branch sections. All governors in the miniWECC were modeled with the tgov1. Further details about the creation and use of the miniWECC may be found in [56], [62], and [28].

The mini WECC was modified from the original configuration to include three areas. The importance of these area definitions becomes more clear in Section 5.4.1. Additionally, since the mini WECC was designed to test heavy loading and transient type events, all loads were reduced by 5%. To keep the voltage profile similar to the original design, reactive shunts were also reduced by 5%.

4.4.3.1 Simulated Scenario Descriptions

Similar to the six machine tests, a load step, load ramp, and generator trip were simulated. At $t = 2$ the loads on buses 16, 21, and 26 were each increased by 400 MW. The ramp perturbation acts on these same loads a total of 400 MW, but over 40 seconds. The ramp is equivalent to forty 30 MW steps taking place every second from $t = 2$ to $t = 42$. Initially generating ≈ 200 MW, generator 27 was tripped at $t = 2$. Unlike the six machine scenario, in which the generator trip was ‘larger’ than the load step; the mini WECC load step is double the magnitude of the simulated generator trip.

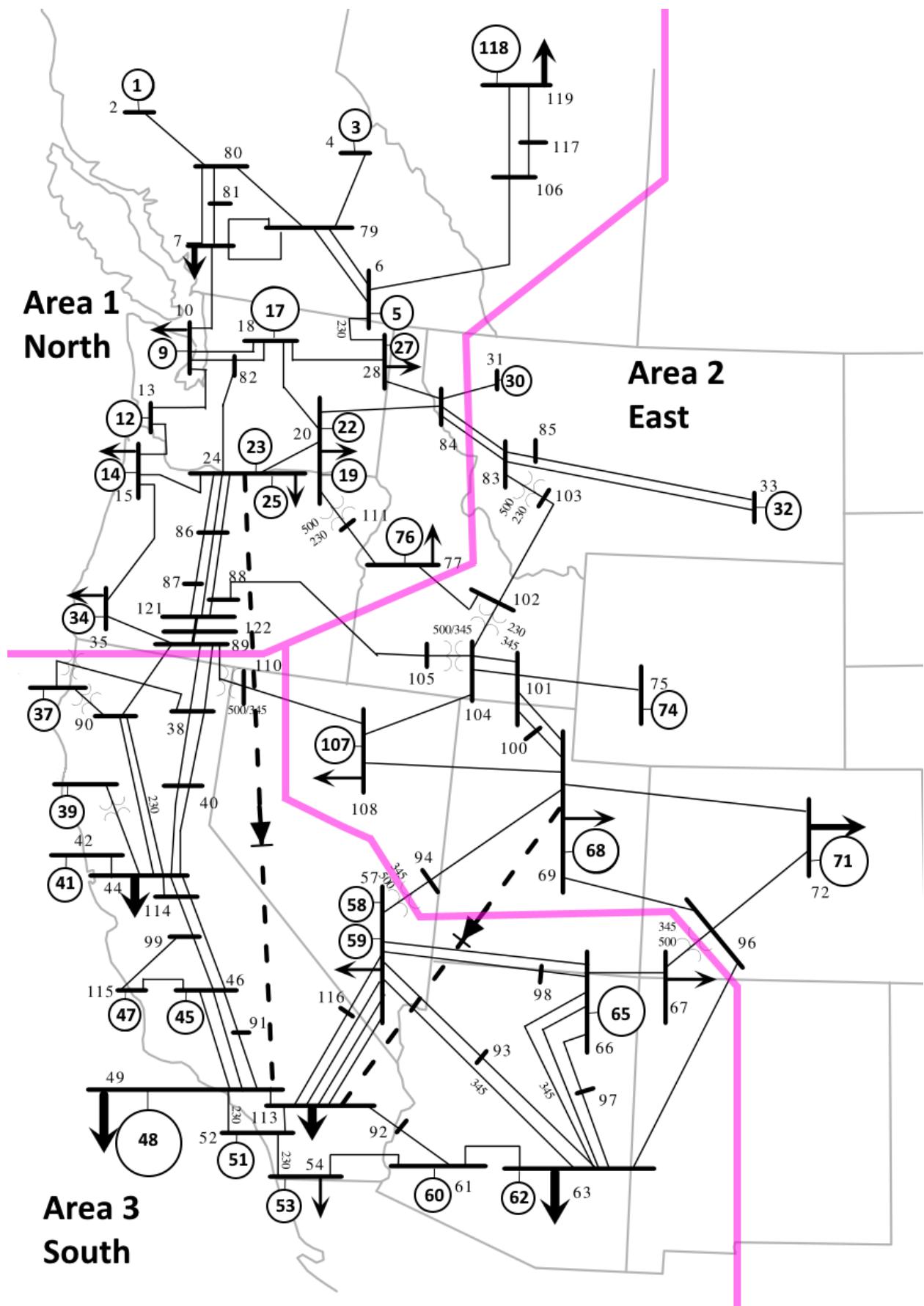


Figure 4.49: Mini WECC system adapted from [28].

4.4.3.2 Frequency Results

System frequency comparison results are presented for each simulated scenario in Figures 4.50, 4.51, and 4.52. The simulated step events caused the PSDS frequencies to oscillate. The generator trip oscillations appear damped while the load step oscillations do not. The weighted system frequency of PSDS appears to follow the center of individual frequency oscillation. Similar to six machine results, the ramp event most closely tracks PSDS behavior. For all cases, absolute frequency difference maximums never exceed 10.0 mHz and are less than 1.0 mHz in the long-term.

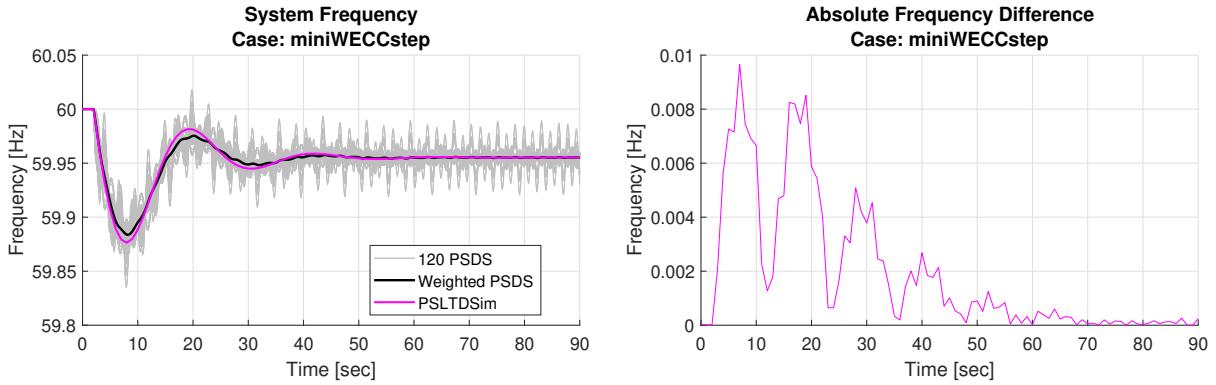


Figure 4.50: Mini WECC load step system frequency comparison.

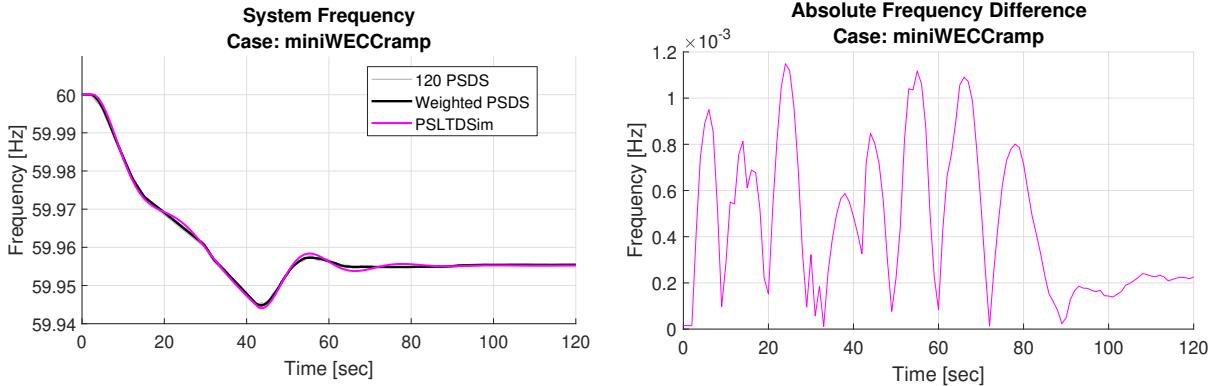


Figure 4.51: Mini WECC load ramp system frequency comparison.

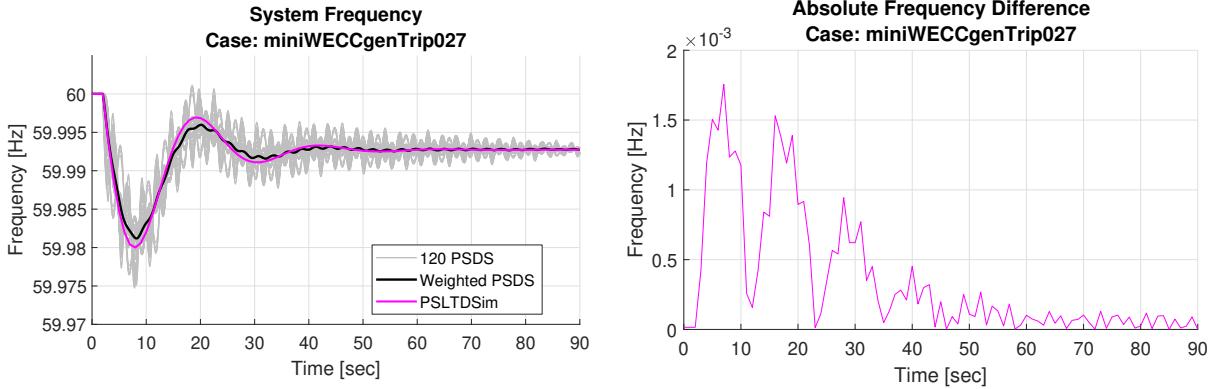


Figure 4.52: Mini WECC generator trip system frequency comparison.

4.4.3.3 Generator Mechanical Power Results

Generator mechanical power comparison results are presented for each simulated scenario in Figures 4.53, 4.54, and 4.55. The ramp and generator trip events both had MW differences of less than 10 MW (1.0% difference), with averages approaching zero in the long-term. The larger load step caused a MW difference of approximately 70 MW (5.0% difference). In both the step type events, the system starts to swing and doesn't completely come to rest by the end of simulated time.

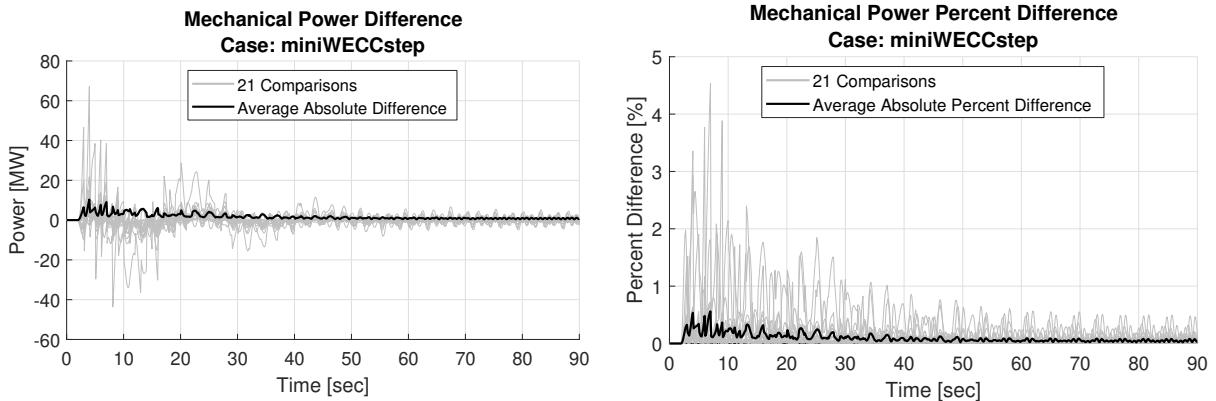


Figure 4.53: Mini WECC load step mechanical power comparison.

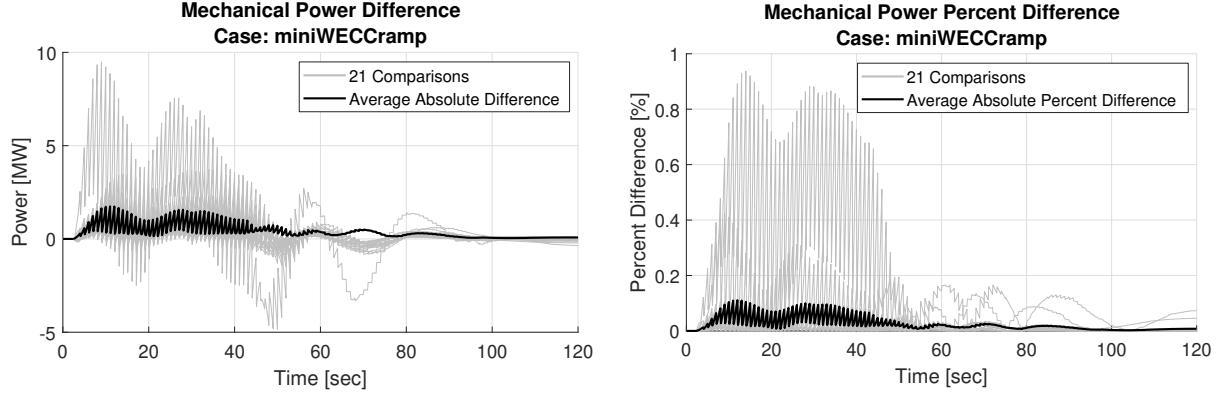


Figure 4.54: Mini WECC load ramp mechanical power comparison.

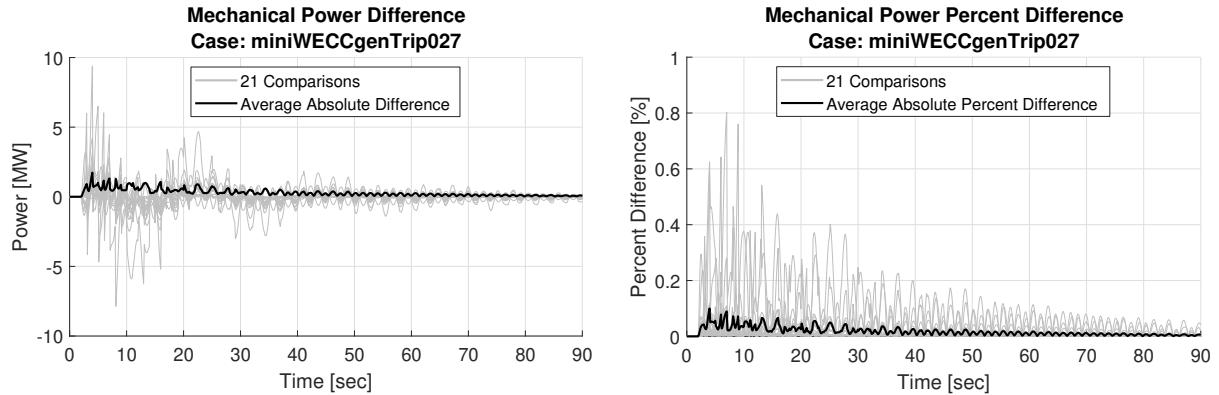


Figure 4.55: Mini WECC generator trip mechanical power comparison.

4.4.3.4 Generator Real Power Results

Generator real power comparison results are presented for each simulated scenario in Figures 4.56, 4.57, and 4.58. The swinging system causes large differences in real power for both step type events, however both absolute average percent differences is less than 5.0%. The ramp event, which differs less than 1.2%, matches PSDS data well. In all scenarios, long-term behavior averages approach zero, or accounting for system swing, approach the center of oscillation.

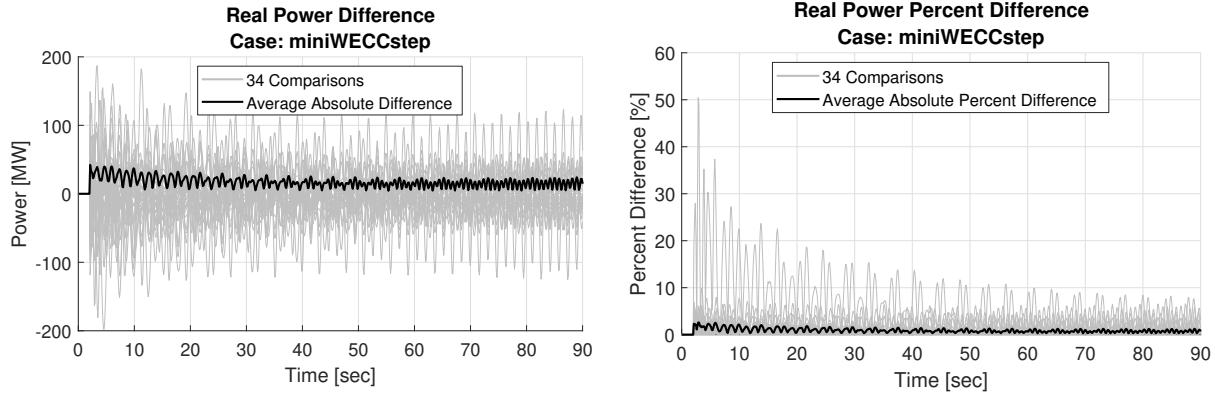


Figure 4.56: Mini WECC load step real power comparison.

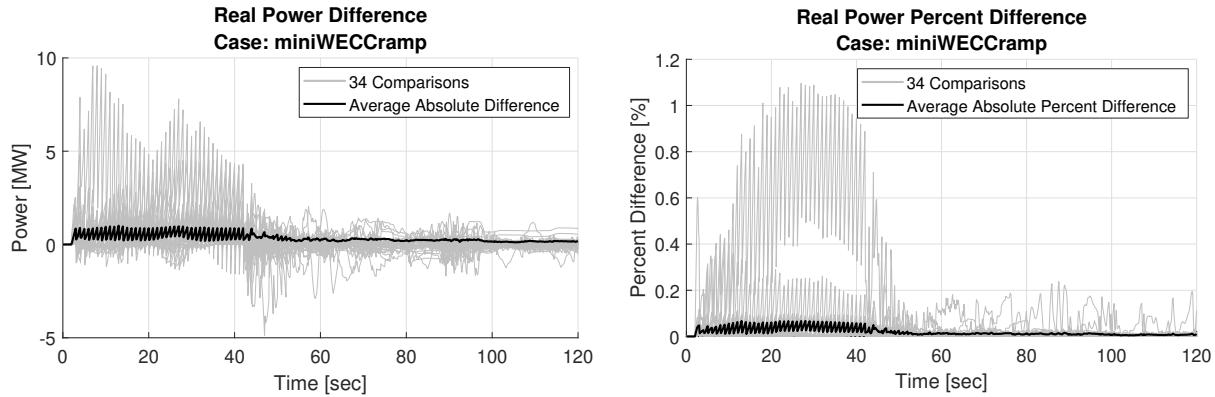


Figure 4.57: Mini WECC load ramp real power comparison.

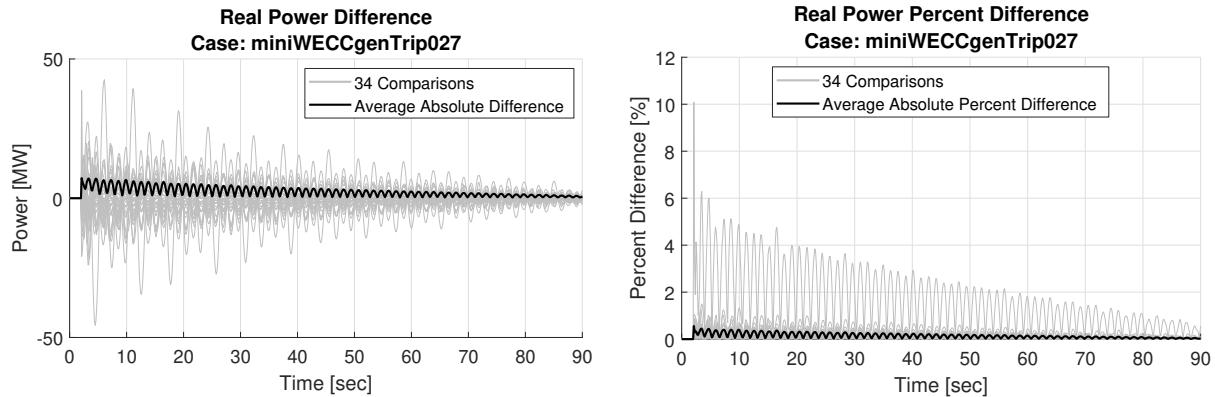


Figure 4.58: Mini WECC generator trip real power comparison.

4.4.3.5 Voltage Magnitude Results

Bus voltage magnitude comparison results are presented for each simulated scenario in Figures 4.59, 4.60, and 4.61. In the load step case, voltage magnitude oscillations appear

to be getting larger as simulated time increases. The generator trip case has damped voltage oscillation behavior. Despite these oscillations, maximum voltage percent difference for either step type event is less than 2.0%. The ramp event has an absolute average percent difference of less than 0.05% for the entire simulation.

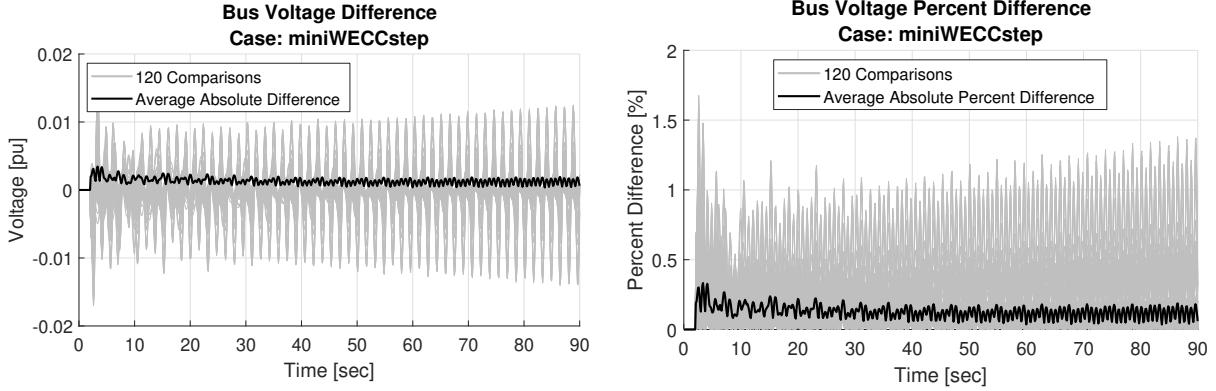


Figure 4.59: Mini WECC load step voltage comparison.

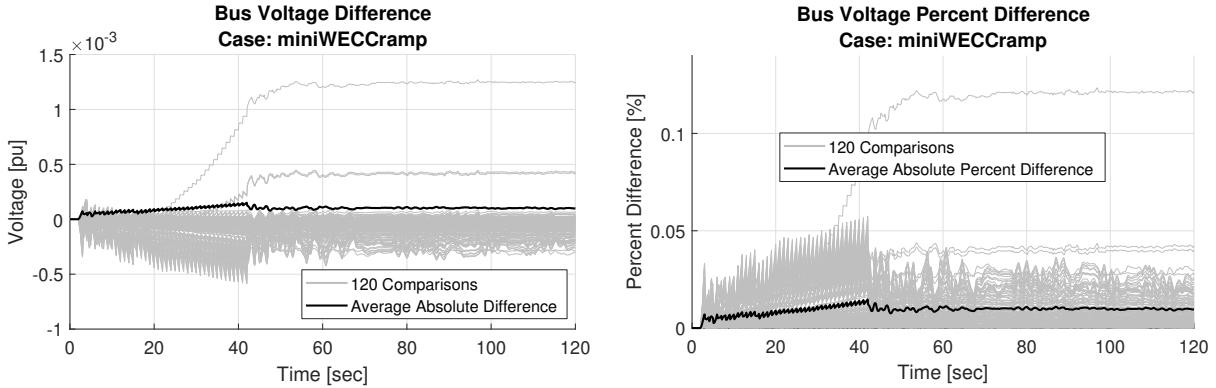


Figure 4.60: Mini WECC load ramp voltage comparison.

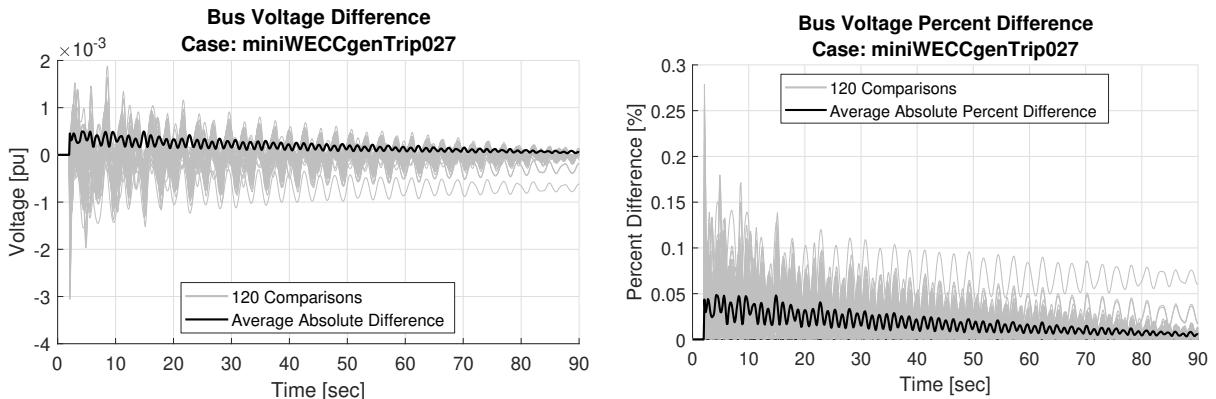


Figure 4.61: Mini WECC generator trip voltage comparison.

4.4.3.6 Voltage Angle Results

Bus voltage angle comparison results are presented for each simulated scenario in Figures 4.62, 4.63, and 4.64. Actual angle differences of less than 10 degrees result in very large percent differences in the load step case. Angle oscillations appear to be damped in the generator trip case and slowly growing in the load step case. The load ramp angle difference magnitude is below 0.3 degrees for the entire simulation. Generally, it can be seen that when the system swing behavior is damped, long-term values from both simulation methods appear to converge.

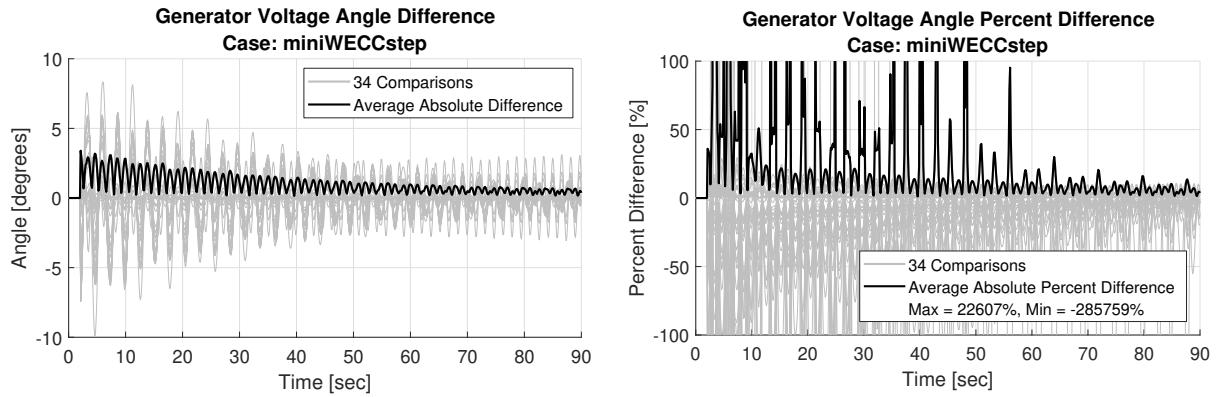


Figure 4.62: Mini WECC load step voltage angle comparison.

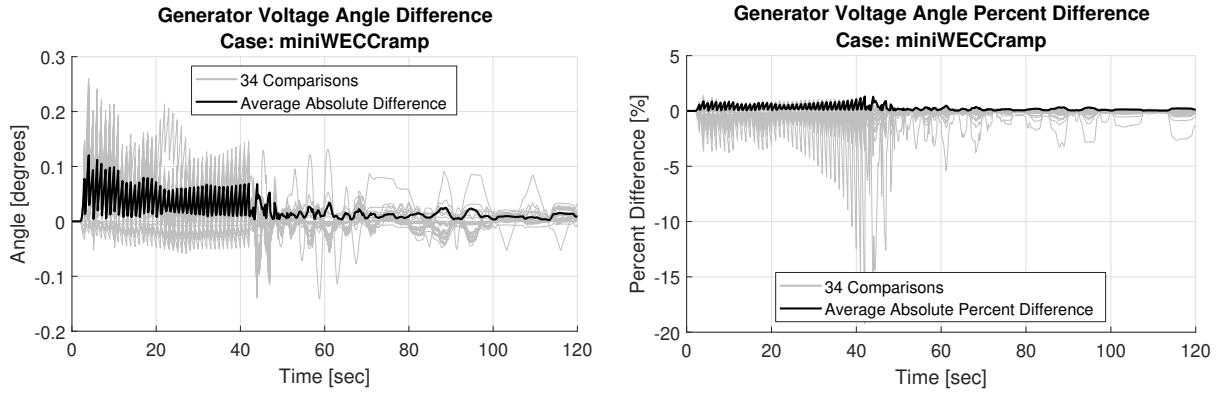


Figure 4.63: Mini WECC load ramp voltage angle comparison.

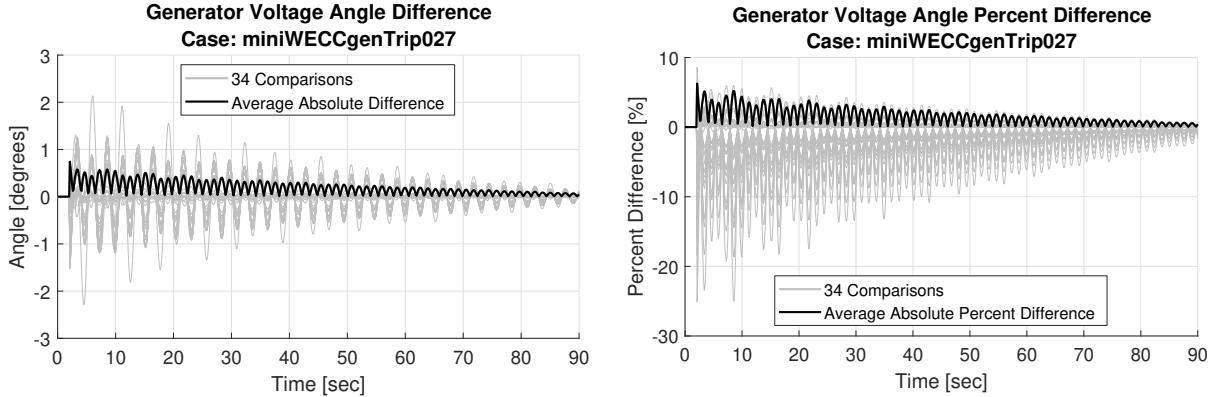


Figure 4.64: Mini WECC generator trip voltage angle comparison.

4.4.3.7 Generator Reactive Power Results

Generator reactive power comparison results are presented for each simulated scenario in Figures 4.65, 4.66, and 4.67. Percent difference plots for both step perturbances oscillate and have multiple large peaks. The ramp case has relatively small MVAR differences and large percent difference spikes at the start and end of the event.

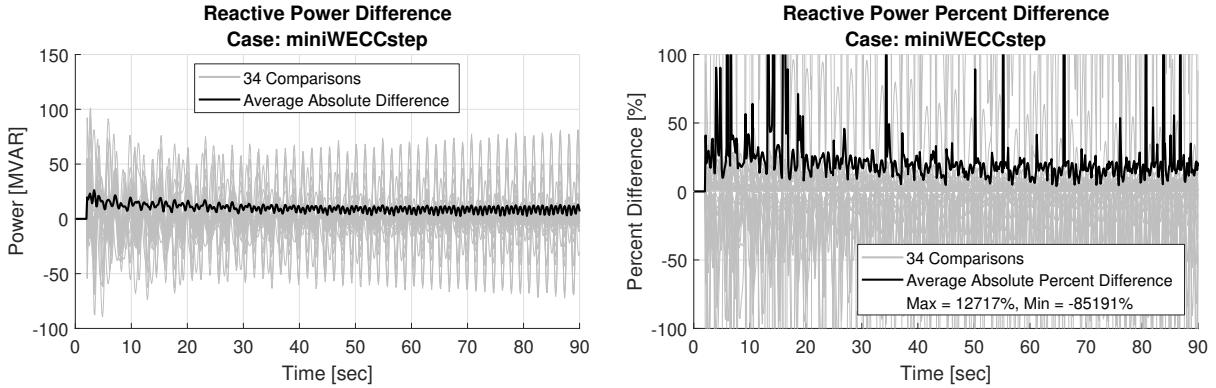


Figure 4.65: Mini WECC load step reactive power comparison.

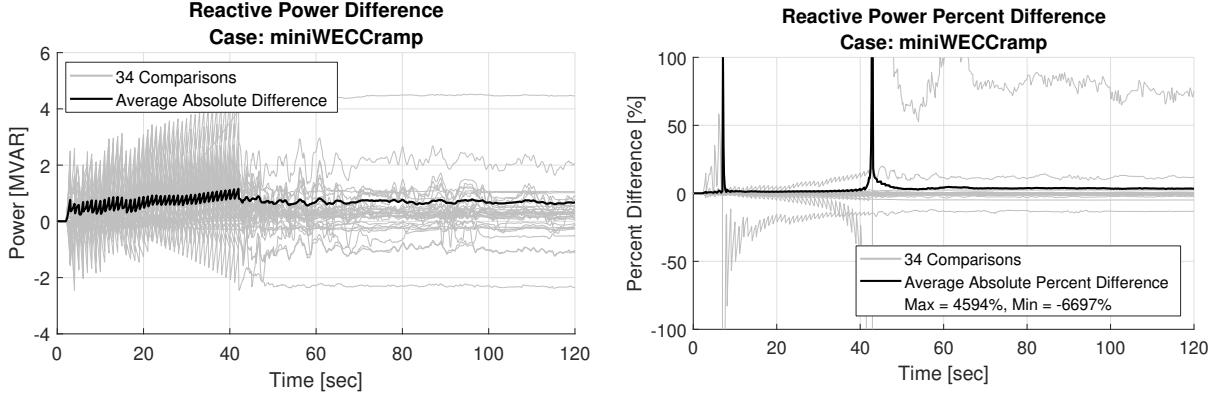


Figure 4.66: Mini WECC load ramp reactive power comparison.

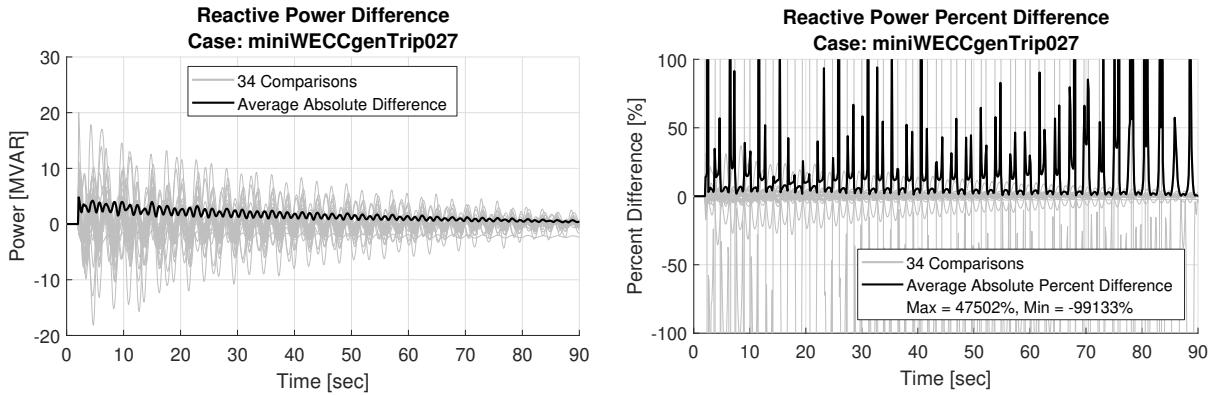


Figure 4.67: Mini WECC generator trip reactive power comparison.

4.4.3.8 Branch Current Results

Branch current comparison results are presented for each simulated scenario in Figures 4.68, 4.69, and 4.70. The ramp scenario has the smallest absolute average percent difference of less than 1.0%. Both step events have relatively low average differences, but fast oscillations are not captured in LTD simulation. Simulations from the generator trip case, which had damped oscillations, appeared to converge in the long-term.

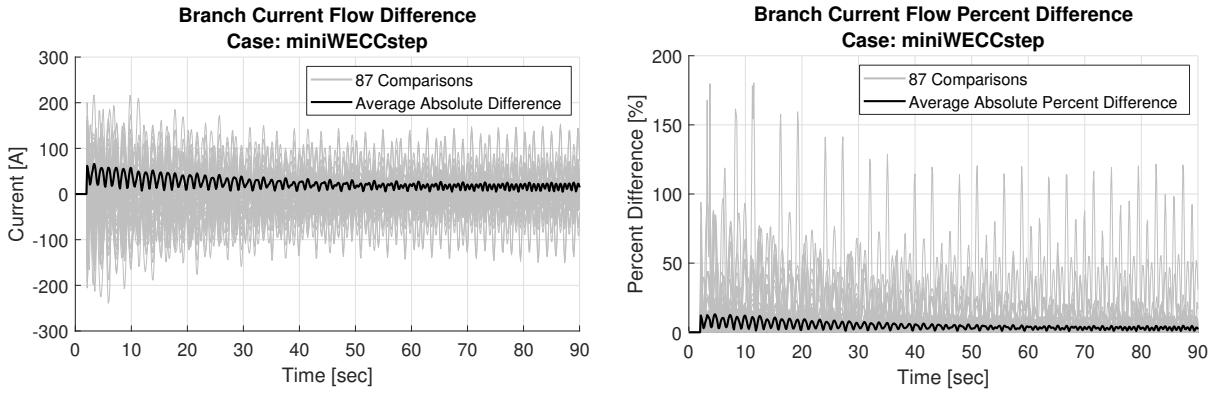


Figure 4.68: Mini WECC load step branch current flow comparison.

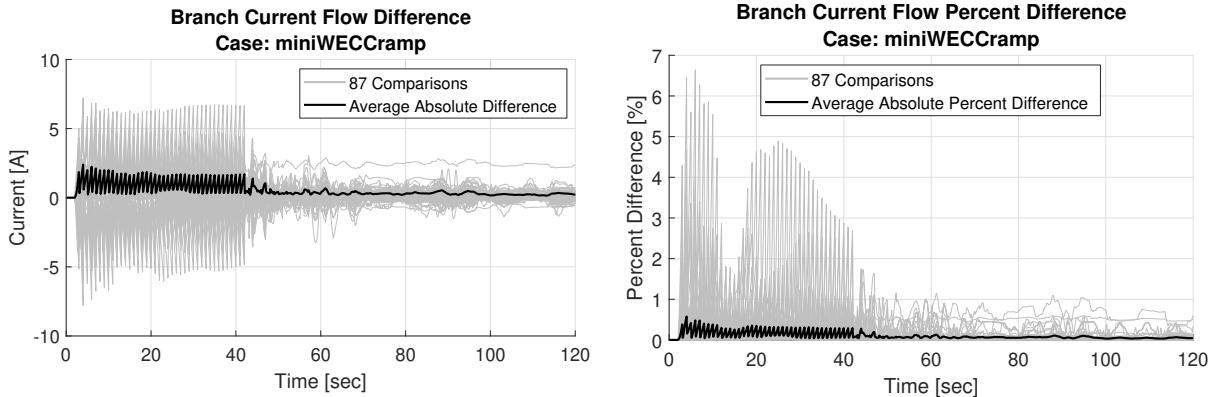


Figure 4.69: Mini WECC load ramp branch current flow comparison.

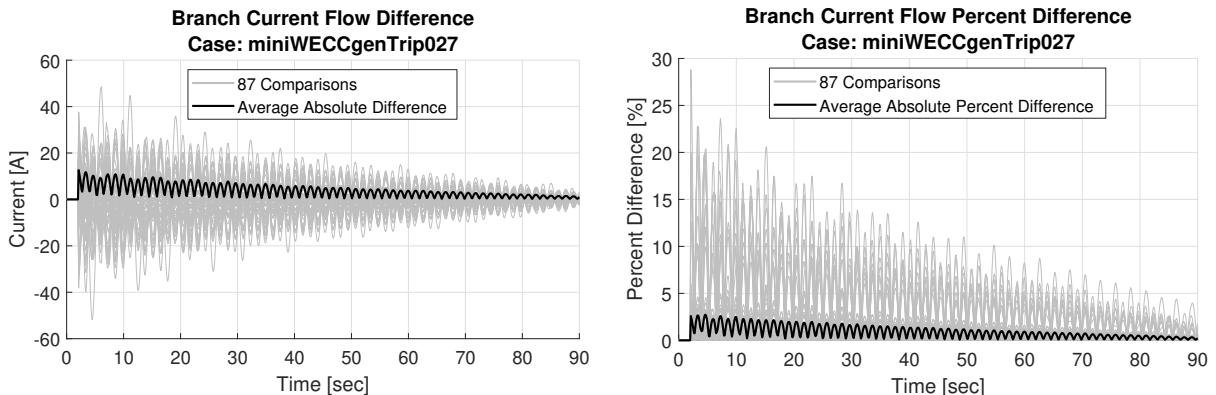


Figure 4.70: Mini WECC generator trip branch current flow comparison.

4.4.3.9 Branch Real Power Flow Results

Branch real power flow comparison results are presented for each simulated scenario in Figures 4.71, 4.72, and 4.73. The load step case results show large peak differences of

nearly ± 100 MW that may be too large for useful simulation analysis. Despite long-term behavior, repeated oscillations present in the generator trip case of over 60% difference may also be too large to be considered acceptable. Ramp results are much more similar to PSDS output with average absolute difference of less than 2 MW.

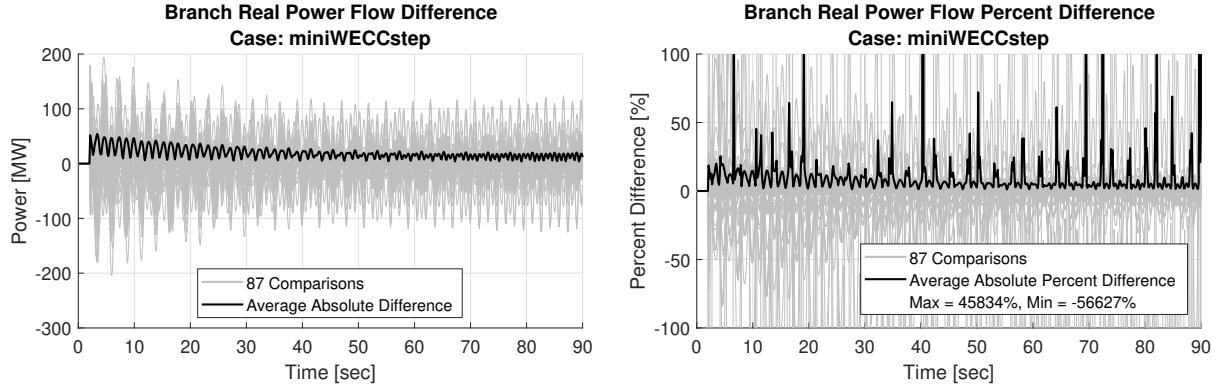


Figure 4.71: Mini WECC load step branch real power flow comparison.

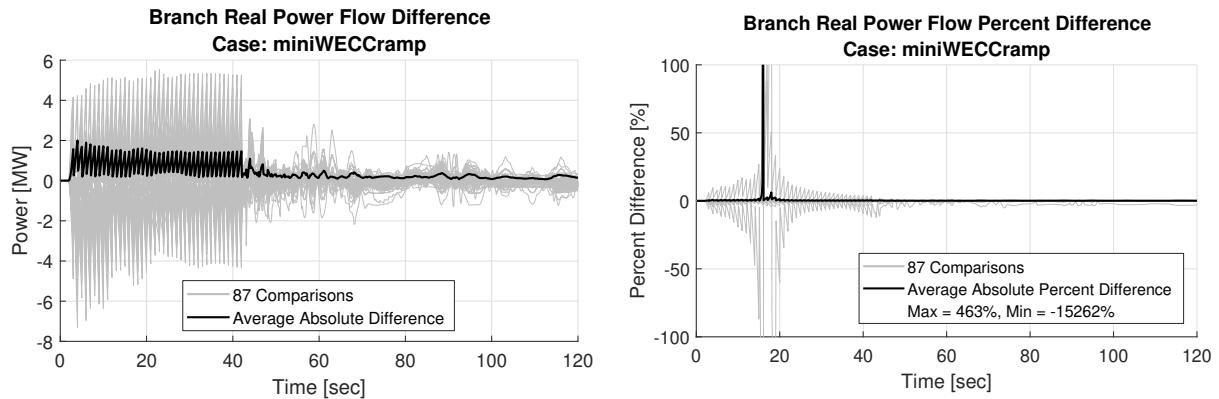


Figure 4.72: Mini WECC load ramp branch real power flow comparison.

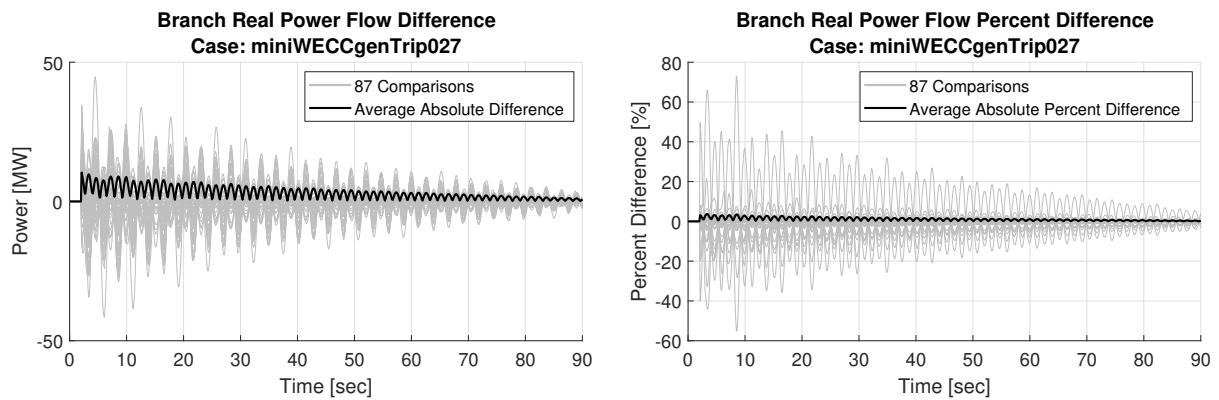


Figure 4.73: Mini WECC generator trip branch real power flow comparison.

4.4.3.10 Branch Reactive Power Flow Results

Branch reactive power flow comparison results are presented for each simulated scenario in Figures 4.74, 4.75, and 4.76. Reactive power flow results all show large percent difference peaks. The load step scenario has growing oscillatory behavior while the generator trip exhibits damped characteristics. Load ramp results differ by only ± 4 MVAR but have a large percent difference.

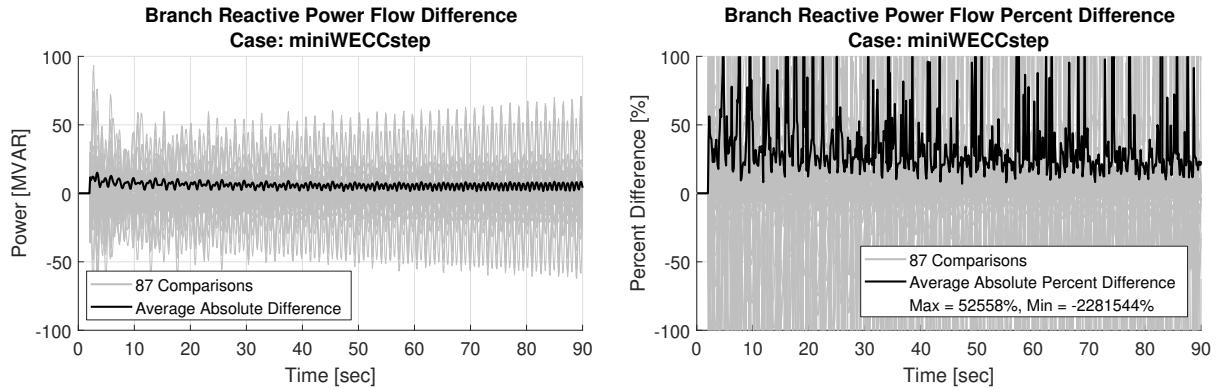


Figure 4.74: Mini WECC load step branch reactive power flow comparison.

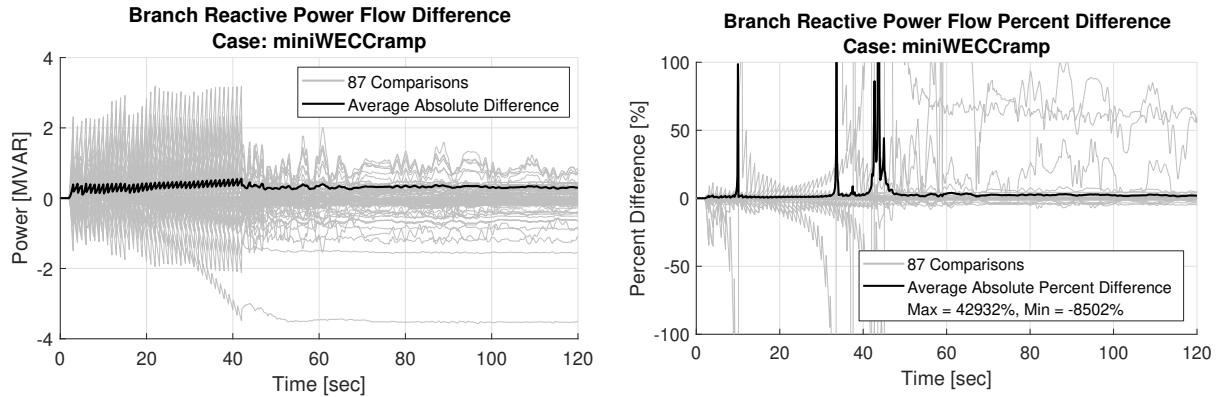


Figure 4.75: Mini WECC load ramp branch reactive power flow comparison.

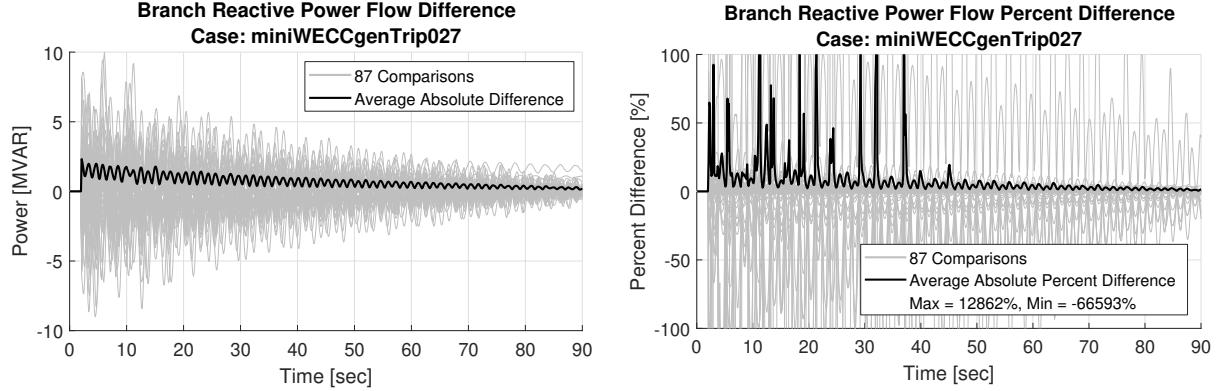


Figure 4.76: Mini WECC generator trip branch reactive power flow comparison.

4.4.3.11 Mini WECC Result Summary

While PSLTDSim assumes a stable system, and a ‘swinging’ system could be thought of as stable, simulation results show large differences in such cases. As such, transient simulations may be a good tool to help decide if a particular scenario is suitable for PSLTDSim. For the mini WECC system, the most suitable event appears to be the load ramp. Undamped voltage oscillations seen in the load step test may indicate that the perturbation was too large for adequate LTD simulation. This hypothesis is further backed up by large differences seen in other results from that particular scenario. Many percent difference plots have very large maximum and minimum peak values. The nature of the percent difference calculation is believed to cause these misleading values during data oscillation, or when compared values are small.

4.4.4 Mini WECC with PSS System

The mini WECC, previously shown in Figure 4.49, was originally created with power system stabilizers (PSS) enabled. To limit the amount of initial modeling differences, and thus assumptions made by PSLTDSim, PSS was removed for initial validation testing. However, step type event results may have been too oscillatory for suitable LTD simulation. Since PSS acts to damp power system oscillations [35], PSS was enabled in PSDS for the following validation tests.

4.4.4.1 Simulated Scenario Descriptions

The same mini WECC simulations from the previous section were conducted to observe PSS behavior and resulting LTD differences. PSDS pss2a models were added to generators on bus 1, 30, 32, 41, and 118. For completeness, perturbances are described again. At $t = 2$ the loads on buses 16, 21, and 26 were each increased by 400 MW. The ramp perturbation acts on these same loads a total of 400 MW, but over 40 seconds. The ramp is equivalent to forty 30 MW steps taking place every second from $t = 2$ to $t = 42$. Generator 27 was tripped at $t = 2$ and was initially generating ≈ 200 MW.

4.4.4.2 Frequency Results

System frequency comparison results are presented for each simulated scenario in Figures 4.77, 4.78, and 4.79. The use of PSS creates a non-steady state operating condition at $t = 0$. This is indicated by the frequency changes that start immediately in all PSDS simulations. Unlike the previous mini WECC load step case, frequency oscillations are damped. General frequency differences are similar to the non-PSS scenario, but ≈ 1.0 mHz larger in each case.

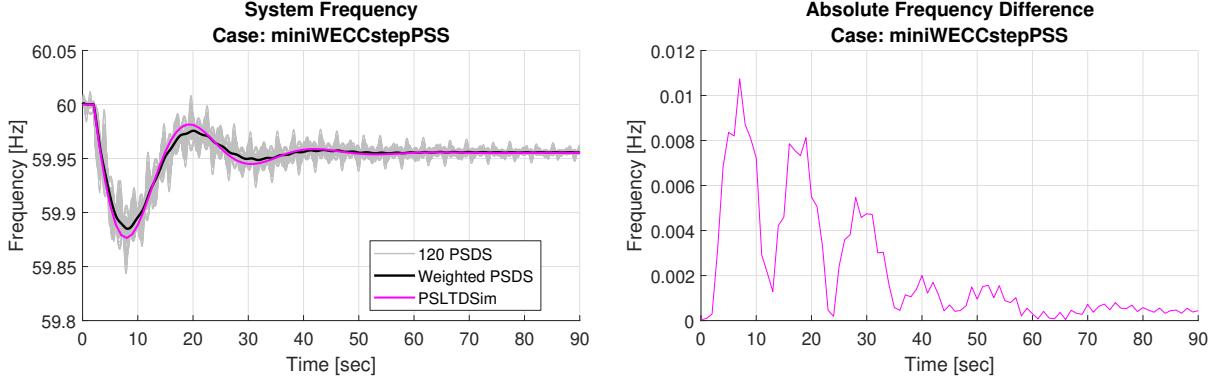


Figure 4.77: Mini WECC with PSS load step system frequency comparison.

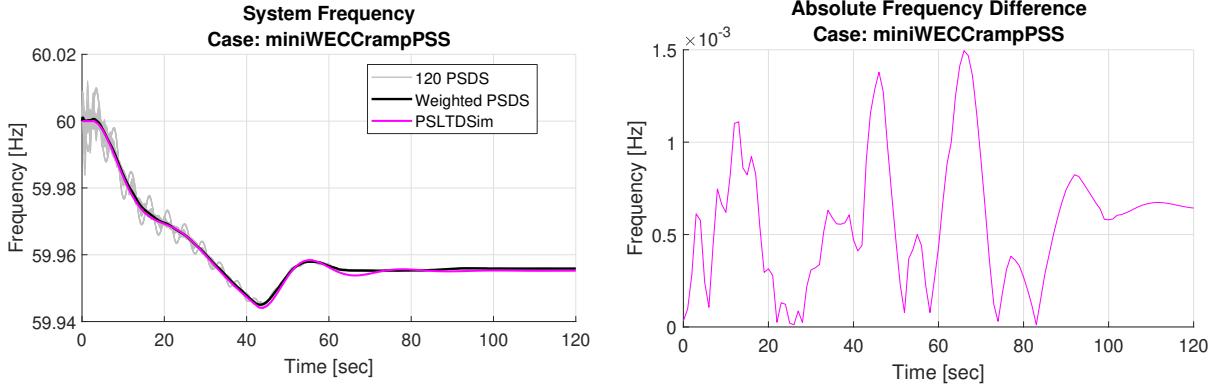


Figure 4.78: Mini WECC with PSS load ramp system frequency comparison.

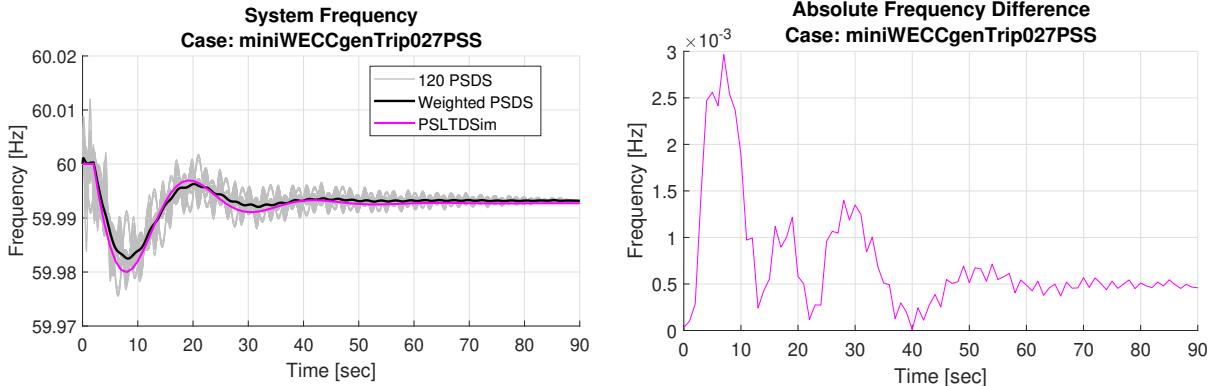


Figure 4.79: Mini WECC with PSS generator trip system frequency comparison.

4.4.4.3 Generator Mechanical Power Results

Generator mechanical power comparison results are presented for each simulated scenario in Figures 4.80, 4.81, and 4.82. Results from the PSS and non-PSS case were mostly the same, but slightly different. The ramp event had smaller oscillatory differences, but

larger steady state differences. The generator trip case also had slightly larger oscillating steady state differences.

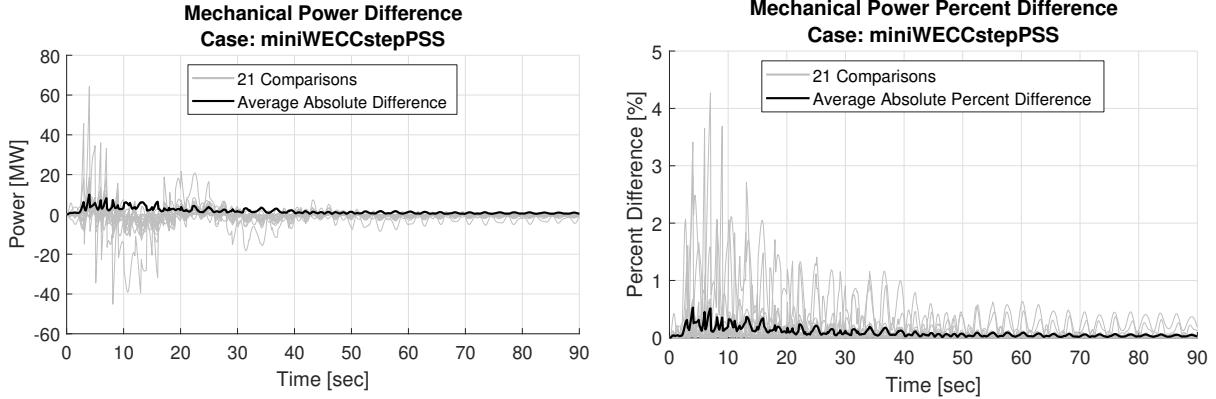


Figure 4.80: Mini WECC with PSS load step mechanical power comparison.

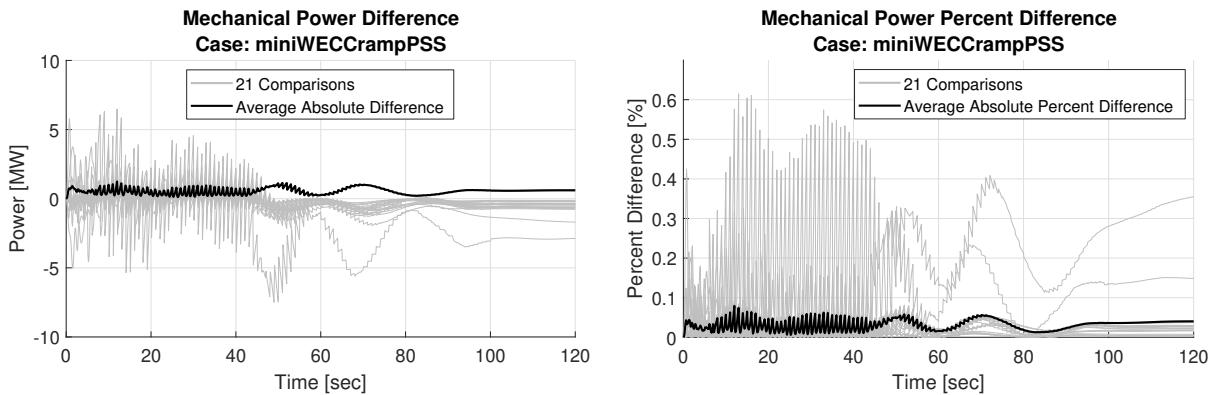


Figure 4.81: Mini WECC with PSS load ramp mechanical power comparison.

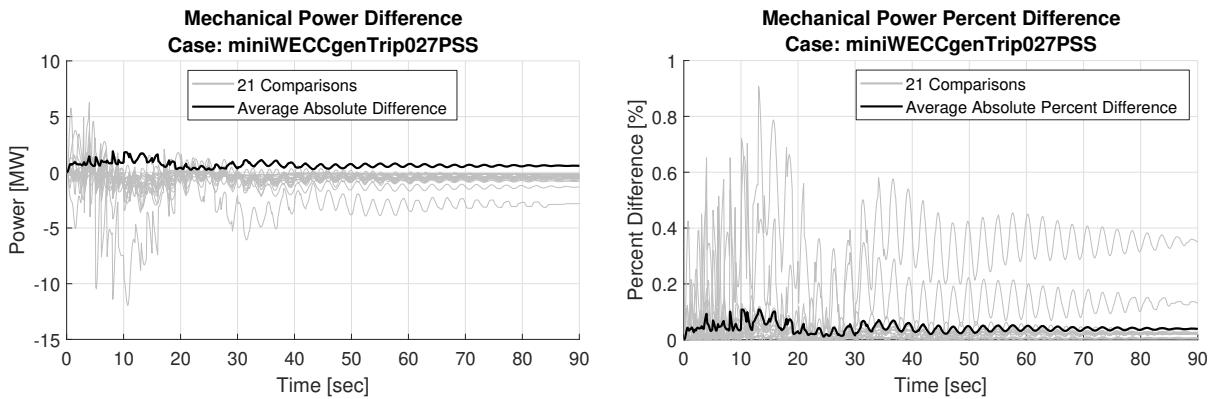


Figure 4.82: Mini WECC with PSS generator trip mechanical power comparison.

4.4.4.4 Generator Real Power Results

Generator real power comparison results are presented for each simulated scenario in Figures 4.83, 4.84, and 4.85. PSS action reduces system oscillation in both step perturbation cases. Peak percent differences are slightly increased from non-PSS results. The ramp validation results appear slightly worse when using PSS because of relatively large initial differences caused by a non-steady state starting condition.

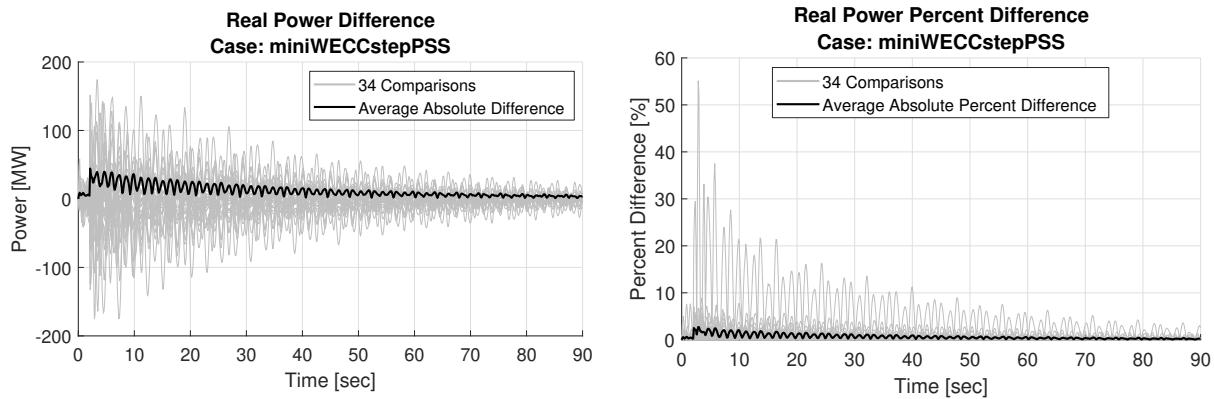


Figure 4.83: Mini WECC with PSS load step real power comparison.

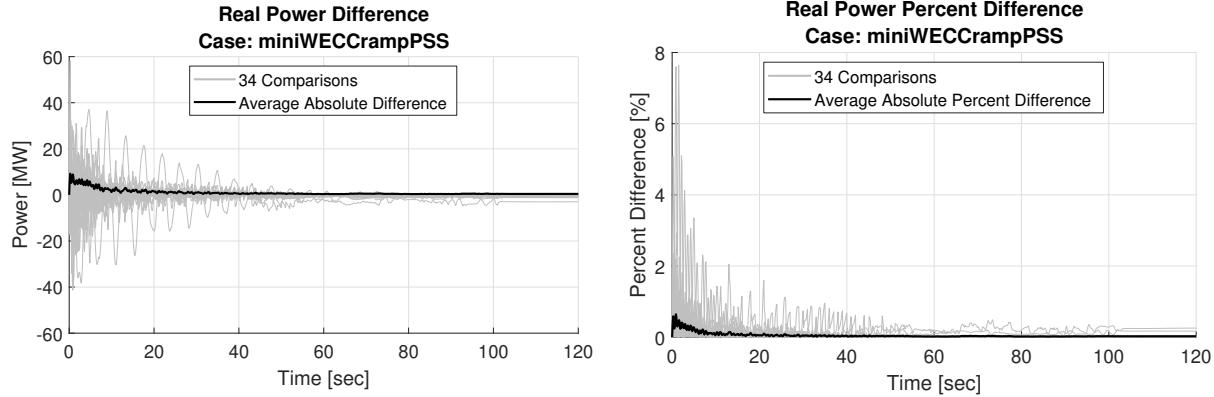


Figure 4.84: Mini WECC with PSS load ramp real power comparison.

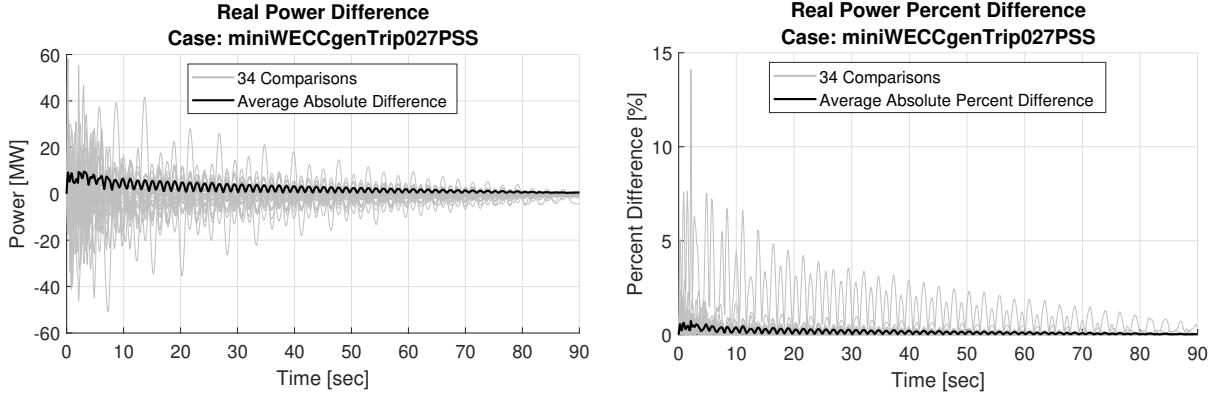


Figure 4.85: Mini WECC with PSS generator trip real power comparison.

4.4.4.5 Voltage Magnitude Results

Bus voltage magnitude comparison results are presented for each simulated scenario in Figures 4.86, 4.87, and 4.88. Long-term voltage magnitudes do not match between simulations. Generally, percent differences peak slightly below 4.0% and do not converge towards zero for all cases. Absolute average percent difference for all cases was slightly below 1.0% for all simulated time.

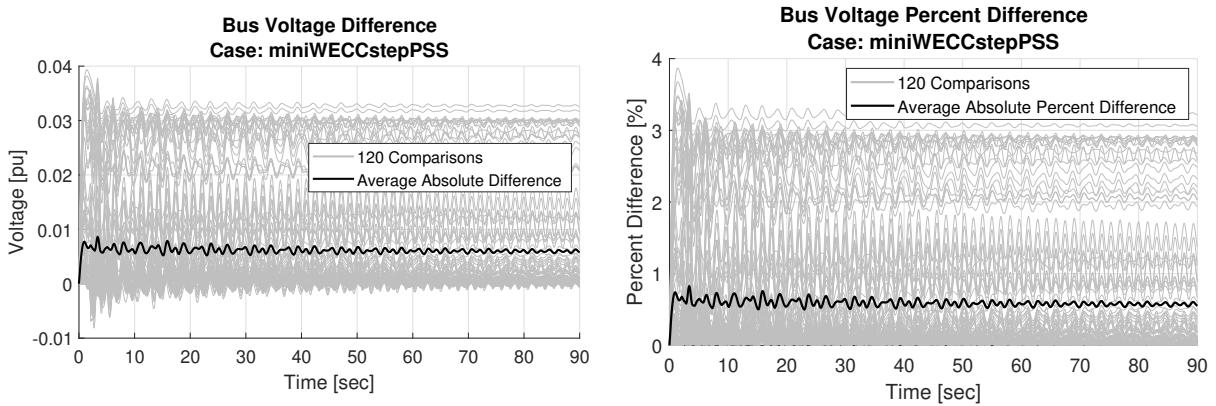


Figure 4.86: Mini WECC with PSS load step voltage comparison.

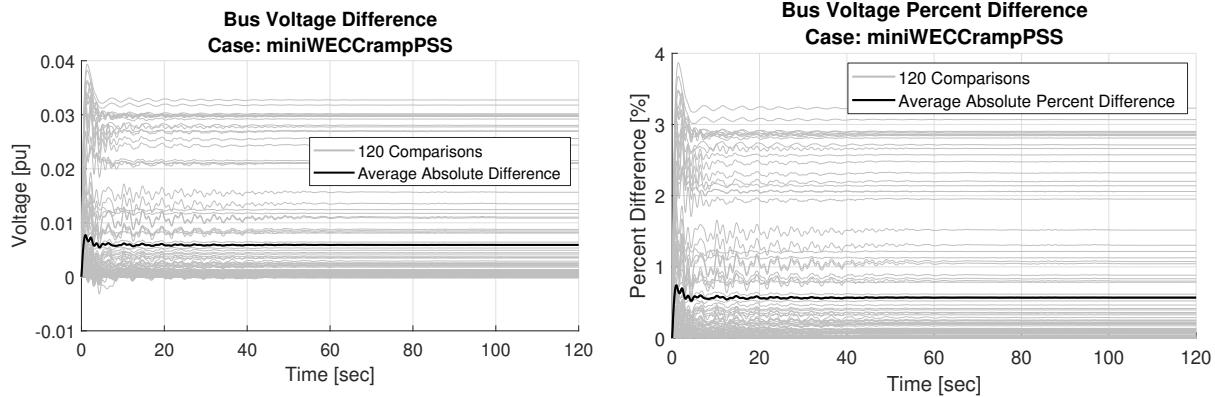


Figure 4.87: Mini WECC with PSS load ramp voltage comparison.

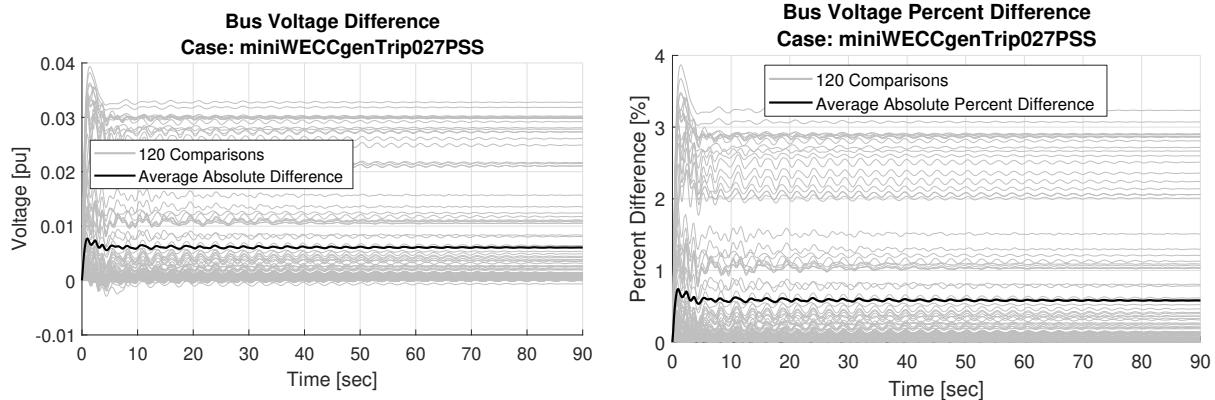


Figure 4.88: Mini WECC with PSS generator trip voltage comparison.

4.4.4.6 Voltage Angle Results

Bus voltage angle comparison results are presented for each simulated scenario in Figures 4.89, 4.90, and 4.91. Step type perturbances with PSS are very similar to non-PSS results. Ramp results show slightly larger angle difference when PSS is used in PSDS.

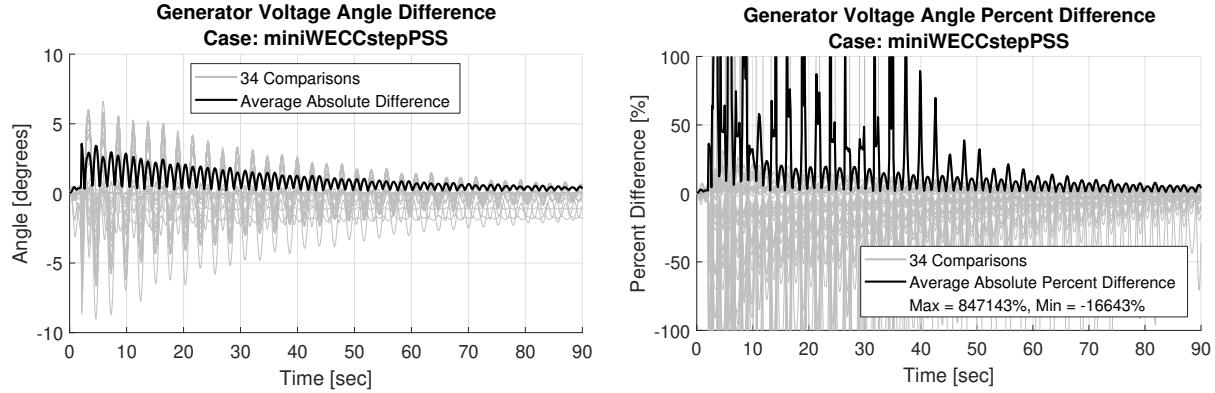


Figure 4.89: Mini WECC with PSS load step voltage angle comparison.

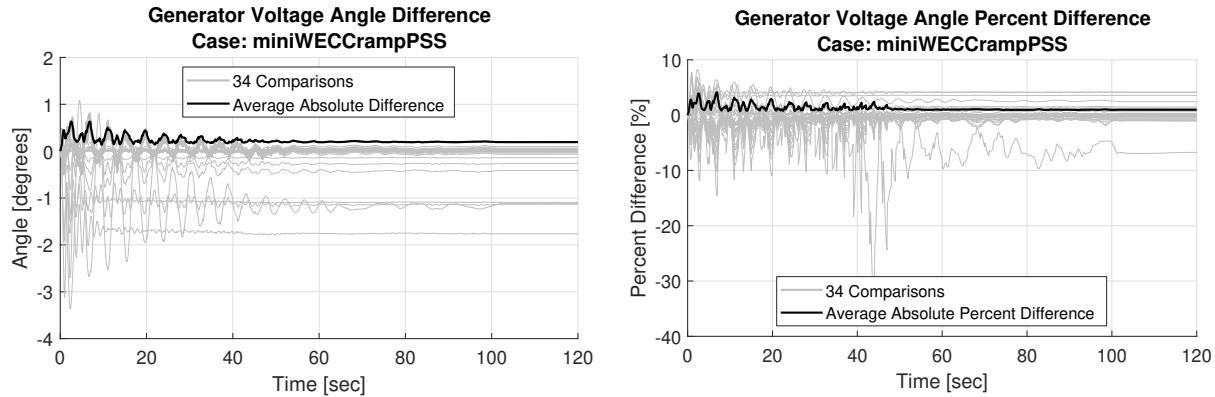


Figure 4.90: Mini WECC with PSS load ramp voltage angle comparison.

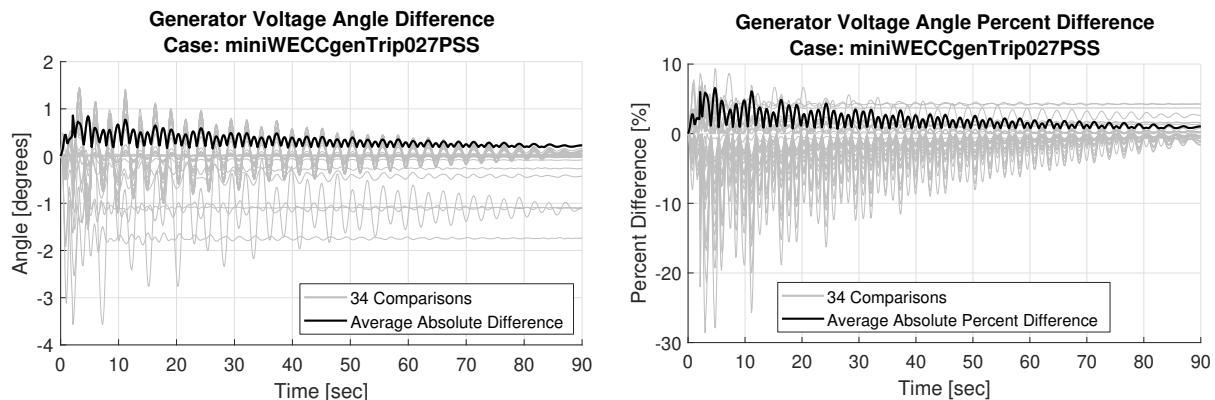


Figure 4.91: Mini WECC with PSS generator trip voltage angle comparison.

4.4.4.7 Generator Reactive Power Results

Generator reactive power comparison results are presented for each simulated scenario in Figures 4.92, 4.93, and 4.94. Using PSS creates large modeling differences in reactive power

output. All test cases have an average absolute difference of 50.0% or larger.

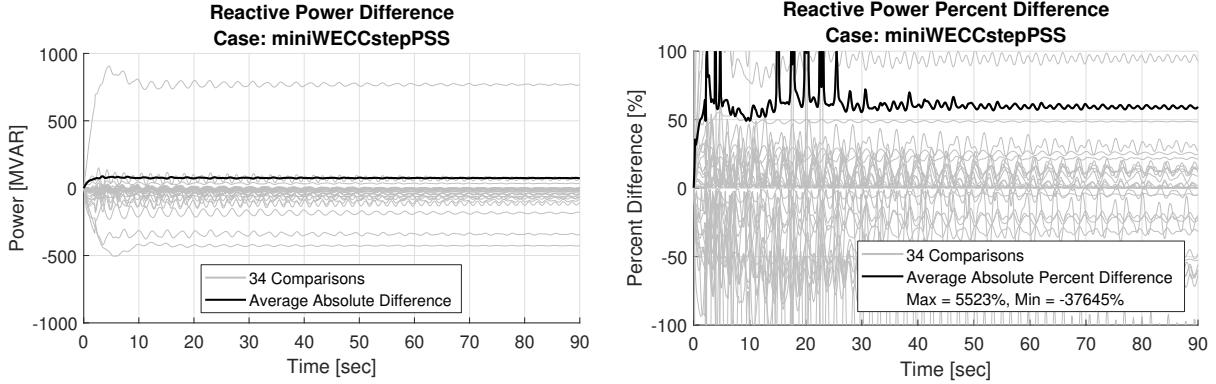


Figure 4.92: Mini WECC with PSS load step reactive power comparison.

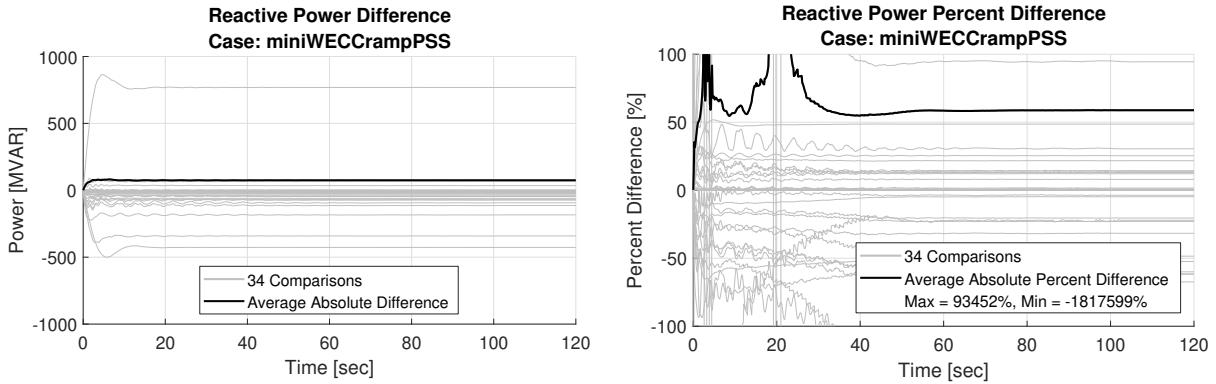


Figure 4.93: Mini WECC with PSS load ramp reactive power comparison.

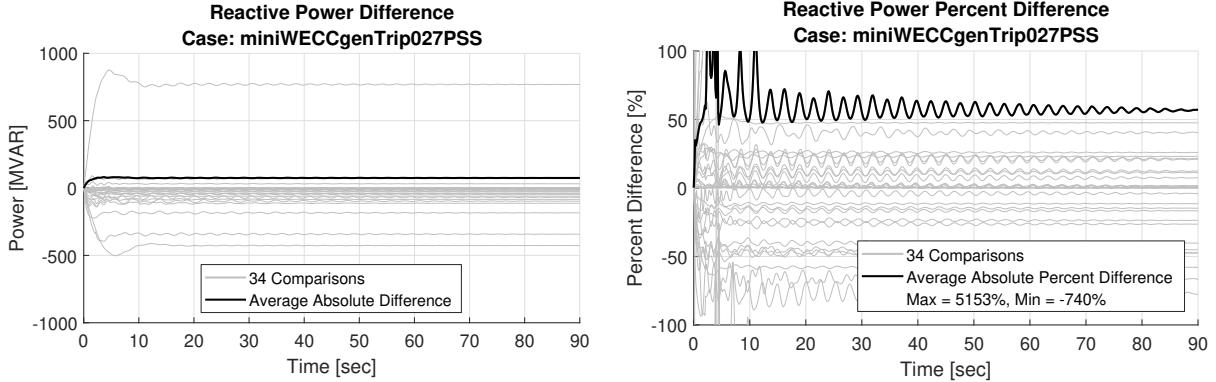


Figure 4.94: Mini WECC with PSS generator trip reactive power comparison.

4.4.4.8 Branch Current Results

Branch current comparison results are presented for each simulated scenario in Figures 4.95, 4.96, and 4.97. Load step results have a damped behavior when PSS is enabled and

slightly smaller peak differences compared to the non-PSS mini WECC results. Absolute percent differences appear similar for all cases. When using PSS, branch current variances during the ramp event increase peak percent differences by nearly five times compared to the non-PSS results.

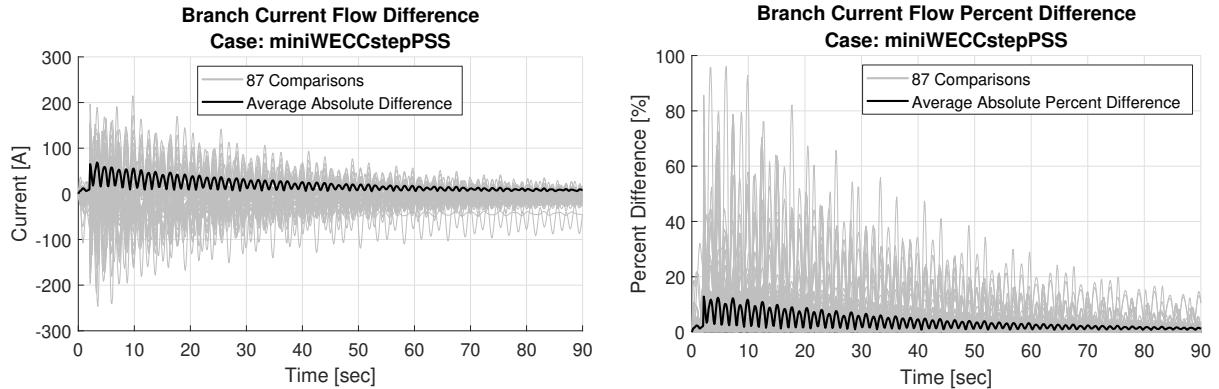


Figure 4.95: Mini WECC with PSS load step branch current flow comparison.

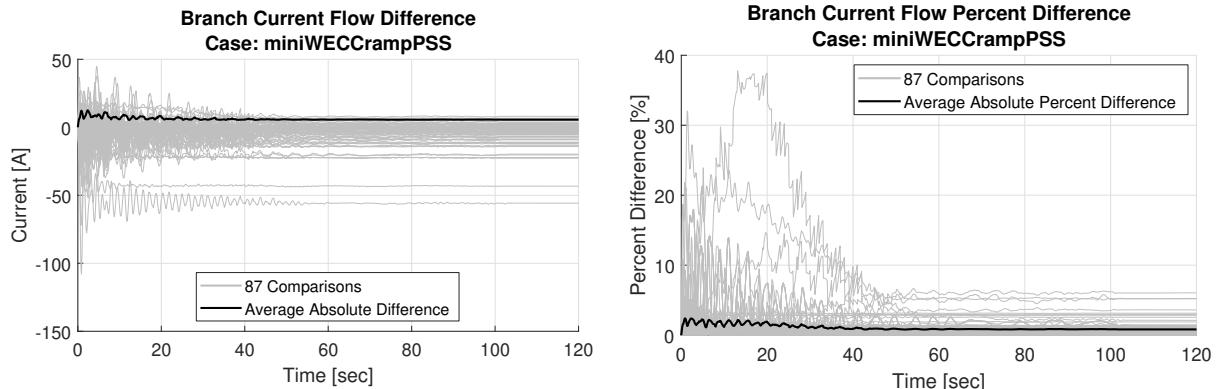


Figure 4.96: Mini WECC with PSS load ramp branch current flow comparison.

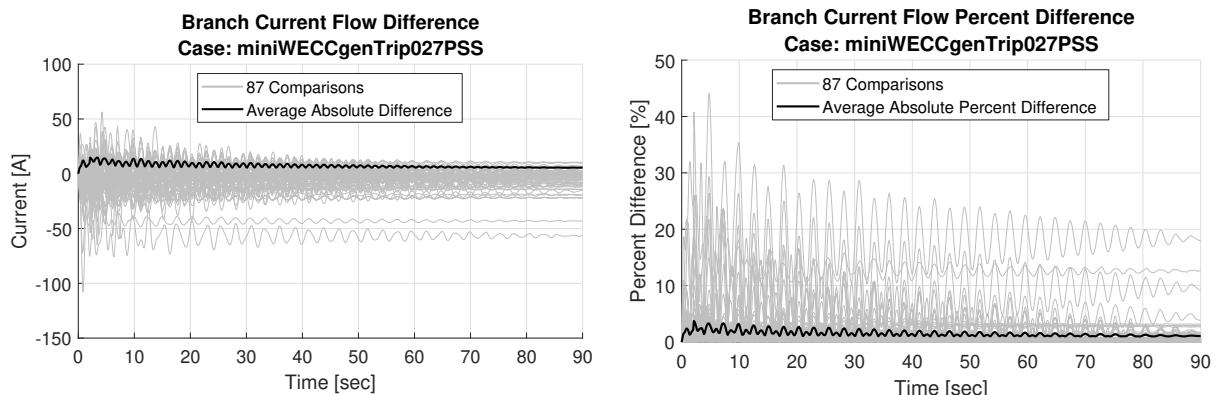


Figure 4.97: Mini WECC with PSS generator trip branch current flow comparison.

4.4.4.9 Branch Real Power Flow Results

Branch real power flow comparison results are presented for each simulated scenario in Figures 4.98, 4.99, and 4.100. With the exception of damped load step behavior, similar results are produced in the step events when PSS is not used. Ramp results appear to have more differences near the beginning of the simulation, but differing scales make further comparisons difficult.

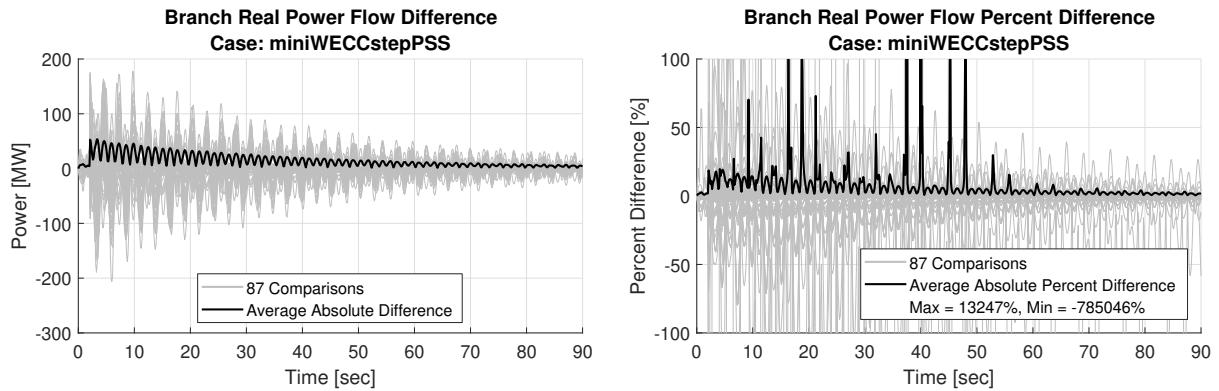


Figure 4.98: Mini WECC with PSS load step branch real power flow comparison.

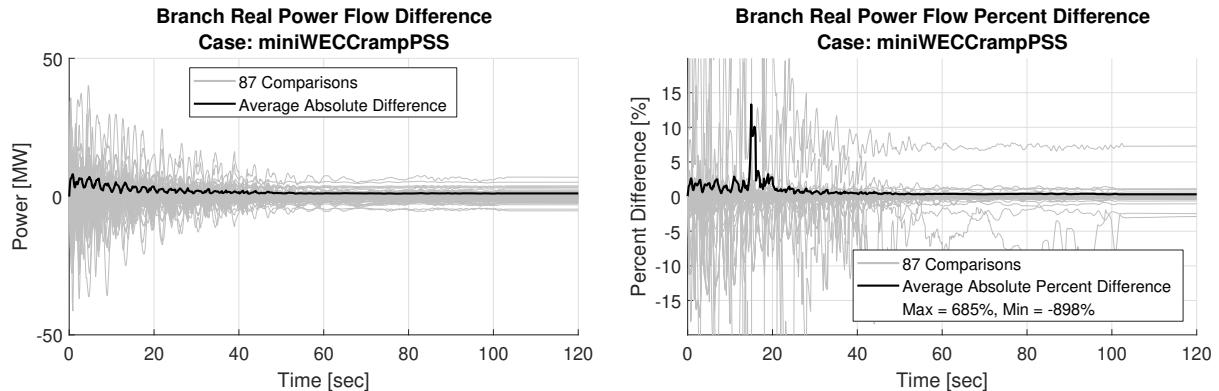


Figure 4.99: Mini WECC with PSS load ramp branch real power flow comparison.

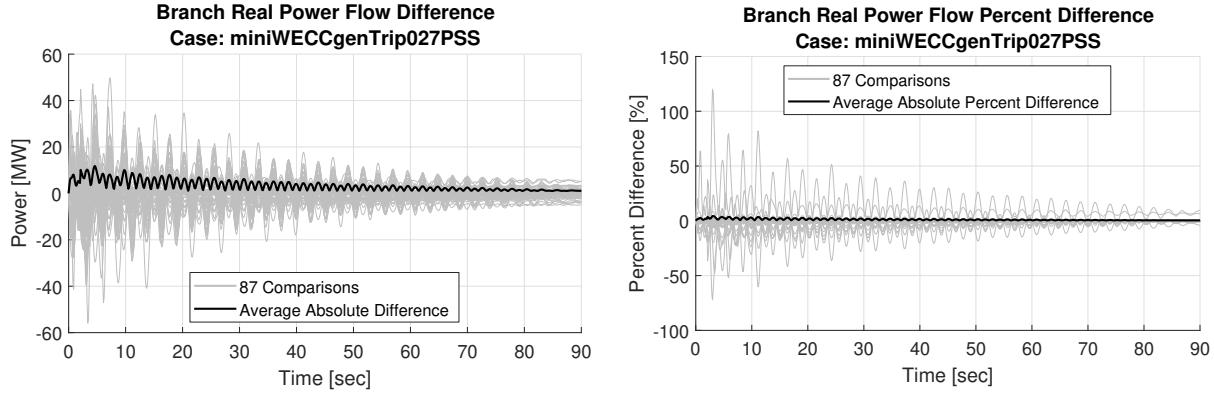


Figure 4.100: Mini WECC with PSS generator trip branch real power flow comparison.

4.4.4.10 Branch Reactive Power Flow Results

Branch reactive power flow comparison results are presented for each simulated scenario in Figures 4.101, 4.102, and 4.103. As should be assumed from the previously noted reactive power differences, reactive power flows also differ by over 50.0% from non-PSS results.

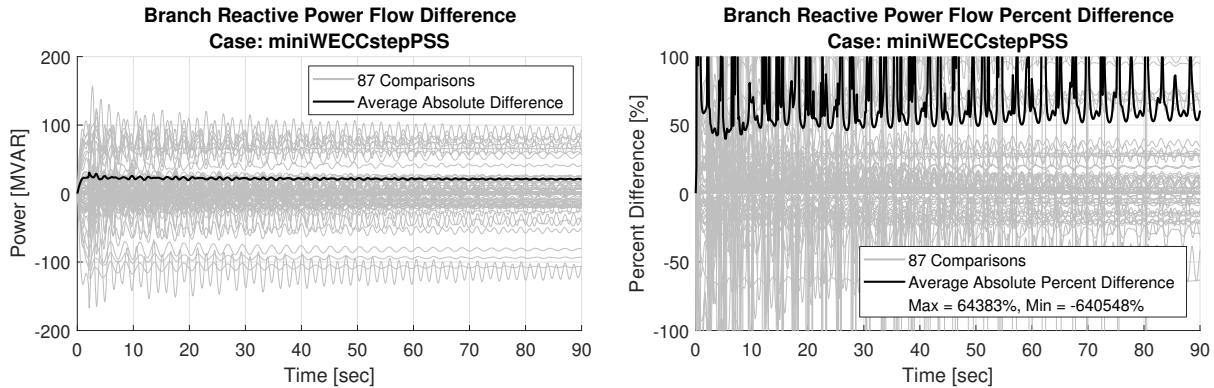


Figure 4.101: Mini WECC with PSS load step branch reactive power flow comparison.

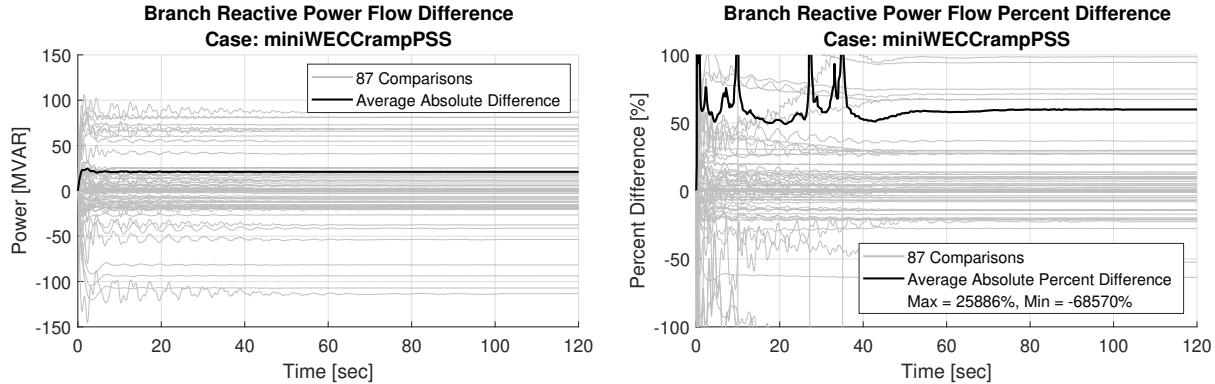


Figure 4.102: Mini WECC with PSS load ramp branch reactive power flow comparison.

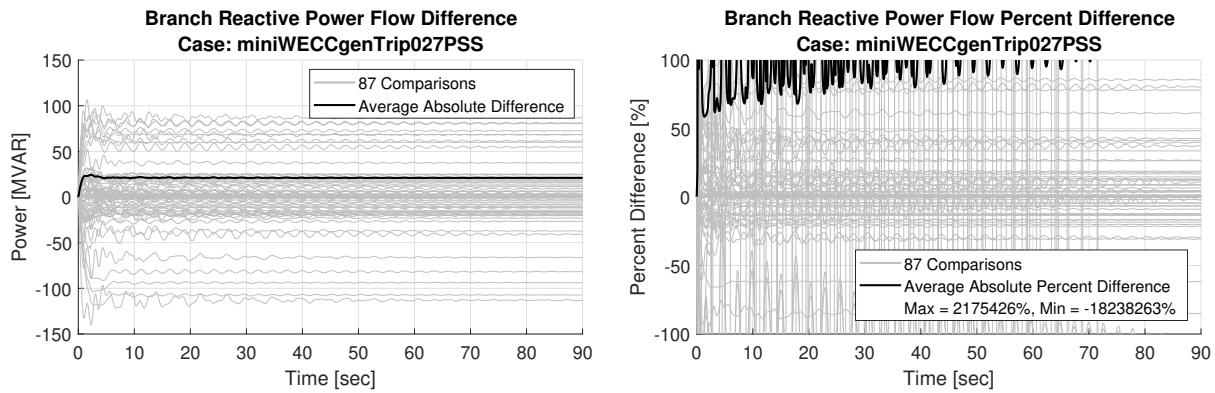


Figure 4.103: Mini WECC with PSS generator trip branch reactive power flow comparison.

4.4.4.11 Mini WECC with PSS Result Summary

The addition of PSS accomplishes the desired goal of stabilizing a power system. However, the additional modeling differences between PSDS and PSLTDSim create more differences in output data. Larger transient differences are seen in most cases as PSS acts to stabilize the system. The pss2a model output is a generator voltage in PSDS which in turn affects steady state voltage magnitudes and branch flow results. Further, initial conditions for all simulations are not the same between PSDS and PSLTDSim as PSS begins to act immediately.

4.4.5 Full WECC System

To test the scalability of PSLTDSim, a full WECC system model was simulated. The full WECC model used is representative of the western United States interconnect during a heavy spring of 2018. The main island consists of 22 areas with a total 21,528 buses and has 3,330 generators operating near an average of 59.3% capacity to supply 140,189 MW of real power to 11,018 loads. A count of modeled generators is shown in Table 4.5. As a reminder, only MW cap, machine base, and machine inertia H are collected from the PSDS model for use in PSLTDSim.

Table 4.5: WECC machine model count and capacity.

Model	Count	Capacity [MW]
gentpj	1,541	96,559.72
genrou	1,277	128,990.75
gentpf	476	17,707.11
motor1	34	717.89
genwri	2	191.00
Total	3,330	244,166.46

The generic governor is used almost exclusively for long-term simulation due to the system model using a wide variety of governor models that are not presently available in PSLTDSim. Table 4.6 shows the total number of PSDS governor models and their respective capacity while Table 4.7 shows the resulting generic model casting performed for long-term simulation in PSLTDSim. It should be noted that about 0.89% of governor models (representing $\approx 0.56\%$ of governed capacity) have the same governor model and time constants as those in PSLTDSim and PSDS due to the generic casting process.

Table 4.6: WECC governor models used in PSDS.

Model	Count	Capacity [MW]
ggov1	1,006	74,804.11
hyg3	315	28,755.47
hygov	196	8,883.36
ieeeg1	191	49,022.45
hygov4	167	8,044.68
ieeeg3	133	9,174.48
gpwscc	56	3,028.33
pidgov	56	8,034.54
gast	29	1,162.56
ggov3	28	5,010.32
hygovr	25	6,249.37
tgov1	20	1,140.45
g2wscc	18	818.95
ccbt1	3	32.53
Total	2,243	204,161.59

Table 4.7: WECC governor models used in PSLTDSim.

Model	Count	Capacity [MW]
gas	1,090	82,842.76
hydro	777	60,786.37
steam	356	59,392.02
tgov1	20	1,140.45
Total	2,243	204,161.59

4.4.5.1 Load Step

For the WECC system model, only frequency response to a load step is presented as the number of assumptions and sheer volume of data make other comparisons difficult with mostly inconclusive results. The perturbation is a step of three loads up 100 MW at $t = 2$. Figure 4.104 shows frequency of all PSDS generator bus frequencies in grey, an average PSDS generator frequency in black, and the LTD system frequency in magenta.

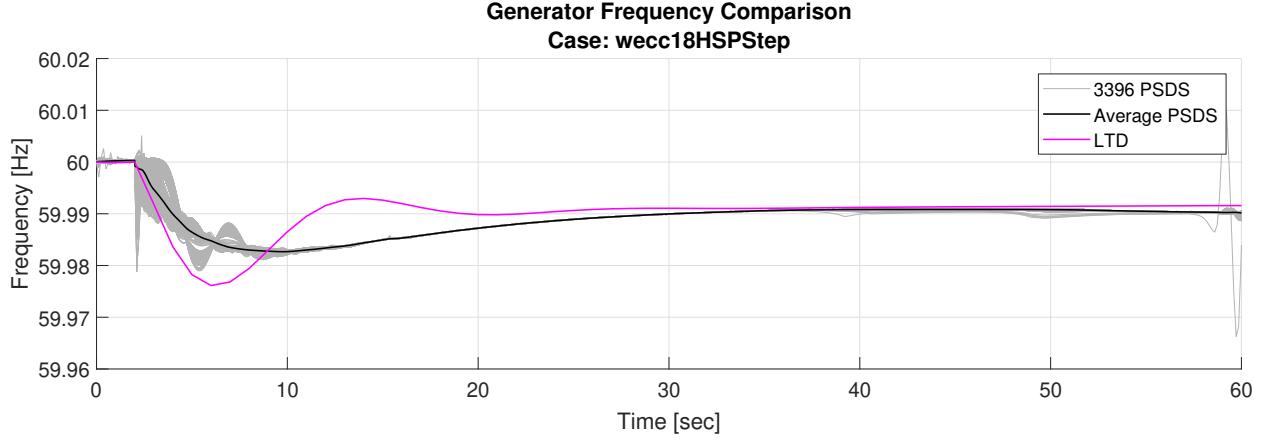


Figure 4.104: Full WECC frequency comparison.

It can be seen that the average PSDS frequency is similar to the LTD frequency immediately after the perturbation. This indicates that the inertial response from the two simulations is similar. However, once governor action becomes more predominate, the two data sets begin to diverge. While the LTD system frequency over estimates the frequency nadir, steady state response is within 2 mHz.

Figure 4.105 shows the absolute frequency difference between PSDS and LTD simulations. Despite governor dynamics being greatly estimated, absolute frequency difference never exceeds 10 mHz. It should be noted that with a perturbation this size, 10 mHz is actually rather large since steady state frequency is about 59.99 Hz.

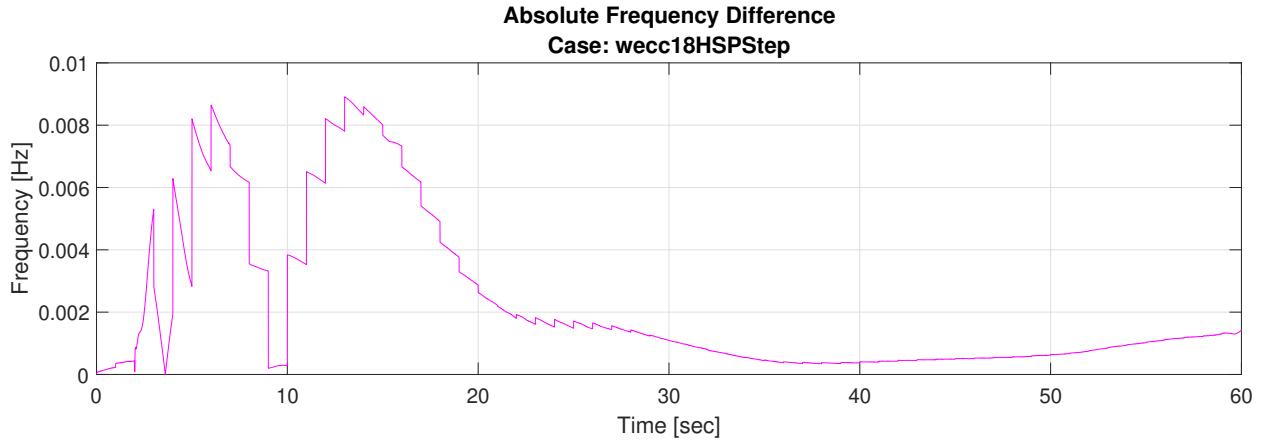


Figure 4.105: Full WECC absolute frequency difference.

4.4.6 Validation Summary

Smaller systems, such as the six machine system, that required less modeling assumptions had generally small differences from PSDS. However, exciter assumptions and the combination of voltage magnitude and angle caused calculated reactive power flow to largely differ from PSDS values during step type events. Oscillations, such as those in found voltage angle, are not well represented in TSPF due to time step resolution.

Larger systems, such as the two WECC models, employ more simplifications in PSLTDSim, and thus have more differences. Large differences are shown in bus voltages and resulting power flow calculations. Ramp events are typically simulated more accurately than step type perturbances. When PSS is used, more voltage behavior is ignored and larger differences are seen in steady-state data. Generic governor casting employed in the full WECC case produced similar steady state behavior, but fairly different dynamic response. System damping, described in Section 5.8, could be used to more accurately match the frequency behavior seen in PSDS.

Overall, PSLTDSim can simulate small or gradual perturbances with acceptable accuracy when compared to PSDS. System frequency response and real power flow results are typically better than voltage and reactive power data. This is more than likely due to the various model simplifications and assumptions in PSLTDSim. Despite transient differences, long-term system behavior tends to converge to PSDS values.

5 Engineering Applications

PSLTDSim may be used to study numerous engineering issues related to power system planning and operation. Due to time and data availability constraints, specific software applications to engineering problems are presented, but specific solutions are not. This chapter contains descriptions of various events of engineering interest that PSLTDSim can simulate, relevant NERC standards for long-term simulation, simulated control options not previously expanded upon, and engineering application studies and results.

5.1 Simulated Events of Interest

The initial focus of PSLTDSim was to study governor and AGC interaction. Reasoning behind this decision was that observed primary response has declined [16], or become slower, while technological advances have enabled AGC action to become faster. The gradual shifting, and resulting overlap, of operation time involved with automatic controls seemed like a good application for LTD simulation. To refine the focus of study, a goal of minimizing system effort while maximizing system stability was developed.

Historically, the number of control pulse signals was used to judge the efficiency of an AGC scheme [7]. Modern control can not be accurately assessed in such a general way. To more reasonably gauge system effort, a metric of governor valve travel was suggested. Valve travel can be greatly affected by various settings such as deadbands and delays. Examples of cumulative valve travel in response to random noise were simulated in PSLTDSim to investigate the effect of certain parameters.

In practice, AGC actions can vary greatly. PSLTDSim offers a BA agent that includes some common AGC techniques. To configure, or tune, an AGC routine requires long-term simulation as gradual system recovery is beyond the transient stability time frame. AGC control of one area affects all connected areas and conflicting control schemes may increase system recovery. An example of AGC tuning is presented and conditional AGC action examined.

To show the ability of PSLTDSim to handle long-term events, a four hour forecast dispatch scenario and a two hour virtual wind ramp were simulated. Generation and load controller agents in PSLTDSim were used to program long duration ramps of system generation and load that use real forecast demand data. During initial simulations, it became apparent that voltage management must be accounted for. Realistically, capacitor switching or other schemes are used to managed voltage and reactive power. In PSLTDSim, a DTC agent was programmed to act as a capacitor bank controller.

Finally, the customizability and open-source nature of PSLTDSim allow for other power system phenomena to be studied. An example of governor action modification using a DTC agent to reproduce an undesired response is presented. Frequency response effects due to system damping and system inertia modification are also demonstrated.

5.2 Relevant NERC Standards

As alluded to earlier, the BES is not operated in any old higgledy-piggledy fashion. NERC has various mandatory standards subject to enforcement in the US that are continuously modified and updated. Standards starting with BAL are related to resource and demand balancing. BAL standards that are applicable to BA simulation are described in this section. As NERC standards change, presented information is predicted to become out of date. The most up-to-date mandatory standards can be viewed on the NERC website [44].

5.2.1 BAL-001-2

NERC standard BAL-001-2 has two requirements. The first requirement deals with a monthly calculation that is out of the scope of PSLTDSim. The second requirement specifies RACE should not exceed a certain frequency trigger limit (FTL) for more than 30 continuous clock-minutes [43]. This requirement is important to BA simulation as PSLTDSim can simulate longer than 30 minutes. NERC provides the following equations describing how to

calculate the balancing authority ACE limit (BAAL).

$$BAAL = -10B(FTL - F_s) \frac{FTL - F_S}{F_A - F_S} \quad (5.1)$$

$$FTL = F_S \pm 3\epsilon \quad (5.2)$$

If actual system frequency F_A is above the scheduled frequency F_S , then the \pm in equation 5.2 is a $+$. If the opposite is true ($F_A < F_S$), then the \pm is a $-$. NERC provides fixed values for each interconnection to use for ϵ . In the WECC, NERC defines ϵ as 0.0228 Hz. Note that minute average frequency is used for F_A the above equations.

Figure 5.1 visualizes the NERC equations for various settings of B . When frequency is at nominal value, ACE is unlimited. Larger values of B equate to larger values of BAAL.

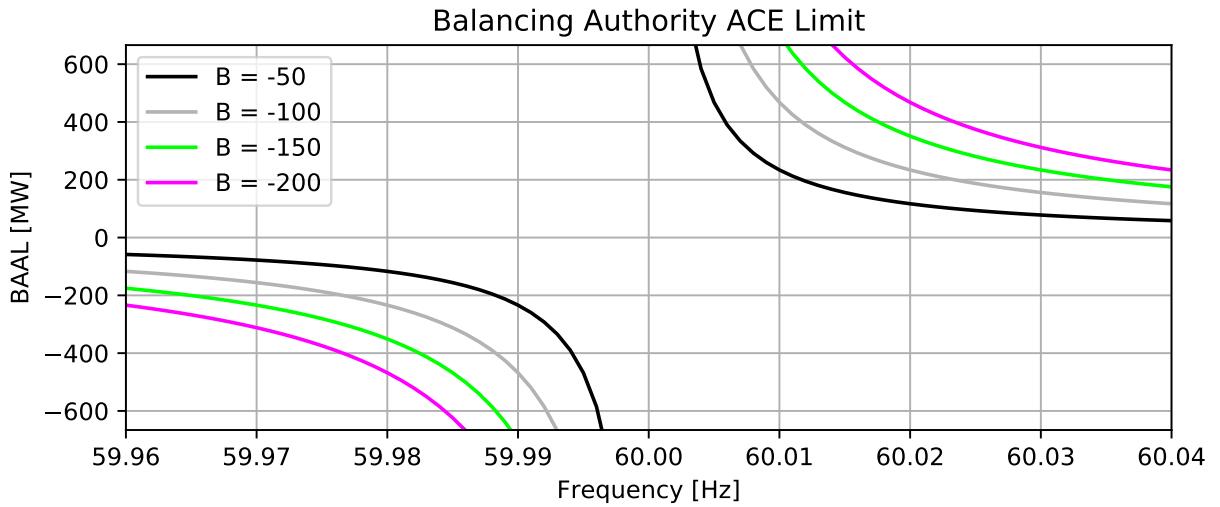


Figure 5.1: Balancing authority ACE limit for different values of B .

5.2.2 BAL-002-3

NERC standard BAL-002-3 requires a BA to return reporting ACE to zero if pre-contingency ACE was positive or zero, or the pre-contingency value if ACE was negative within the contingency event recovery period. The NERC definition for contingency event recovery period is 15 minutes after the start of a contingency [46].

5.2.3 BAL-003-1.1

NERC standard BAL-003-1.1 deals with the frequency bias setting used by a BA. While realistic frequency bias B is calculated based on real system values and recorded responses, this doesn't apply to theoretical models. For simplicity, a setting of 0.9% maximum generation capacity will be used for the minimum fixed value of B . This setting is in line with recommendations made in [42].

5.2.4 NERC Standard Summary

To follow NERC recommendations, a minimum frequency bias B of 0.9% maximum generation capacity will be used for all BAs RACE calculation. A variable frequency bias may be employed for AGC action as this is in-line with practical usage. RACE must return to zero or the pre-contingency level within 15 minutes of a contingency. Since simulated systems start from a steady state, this could be interpreted as frequency crossing zero at least once every 15 minutes. Finally, any control that allows for more than 30 consecutive minutes of RACE exceeding the calculated BAAL will not be acceptable as it violates BAL-002-3.

5.3 Simulated Balancing Authority Controls

Simulated balancing authority controls affect governor response and AGC operation. These controls are defined in the sysBA dictionary in the .ltd.py file. Figure 5.2 shows an example of a BA parameter dictionary as it would appear in a .ltd.py file. A complete list of BA agent options is shown in Table C.1 in Appendix C. The following sections detail most sysBA dictionary keys.

```

1  # Balancing Authority Definition
2  mirror.sysBA = {
3      'BA1':{
4          'Area': 1,
5          'B': "0.9 : permax", # MW/0.1 Hz
6          'AGCActionTime': 30.00, # seconds
7          'ACEgain' : 1.0,
8          'AGCType':'TLB : 4', # Tie-Line Bias
9          'UseAreaDroop' : False,
10         'AreaDroop' : 0.05,
11         'IncludeIACE' : True,
12         'IACEconditional': True,
13         'IACEwindow' : 30, # seconds - size of window - 0 for non window
14         'IACEScale' : 1/5,
15         'IACEdeadband' : 0, # Hz
16         'ACEFiltering': 'PI : 0.04 0.0001',
17         'AGCDeadband' : None, # MW? -> not implemented
18         'GovDeadbandType' : 'nldroop', # step, None, ramp, nldroop
19         'GovDeadband' : .036, # Hz
20         'GovAlpha' : 0.016, # Hz - for nldroop
21         'GovBeta' : 0.036, # Hz - for nldroop
22         'CtrlGens': ['gen 1 : 0.5 : rampA',
23                      'gen 2 1 : 0.5 : rampA',
24                      ],
25     }
26 }
```

Figure 5.2: Single sysBA dictionary definition.

5.3.1 Governor Deadbands

The FERC maximum deadband is 36 mHz[18]. However, the execution of a governor deadband is not explicitly detailed and left to generator operators to configure. PSLTDSim offers a deadband agent that can apply various deadbands to the incoming $\Delta\omega$. Figure 5.3 graphically depicts how the various available deadband options differ.

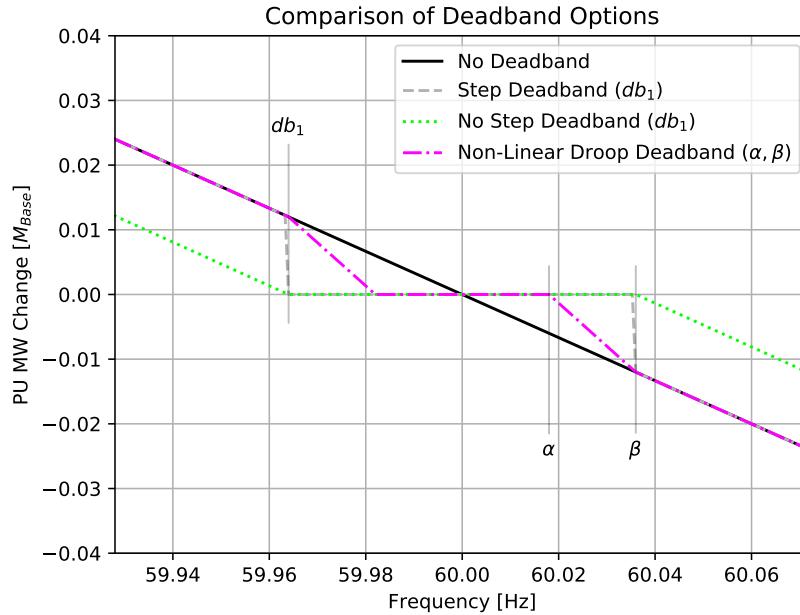


Figure 5.3: Examples of available deadband action.

A step deadband simply nullifies any $\Delta\omega$ whose absolute value is less than the prescribed deadband. A no-step, or ramp, deadband removes the step characteristic by essentially pushing the original droop curve out to deadband thresholds. While the no-step deadband has no abrupt changes, the actual governor response will always be below the ideal droop response. To eliminate this oversight, a non-linear droop option was created that ramps from a set Hz deadband to the ideal droop curve.

Configuring a deadband is done via the sysBA parameter dictionary or the governor deadband dictionary in the .ltd.py file. The dictionary keys are the same in the sysBA dictionary as the govDeadBand dictionary. Details about creating deadbands using a governor

deadband dictionary is presented in Section 4.3.2.3.7.

5.3.2 Area Wide Governor Droops

The typically recommended FERC droop is 5%[18]. Area wide governor droops can be specified in the sysBA dictionary that overwrite any droop setting previously read from a .dyd. All active governors in an area will use the specified ‘AreaDroop’ value if ‘UseAreaDroop’ is True. This setting allows for fast and easy configuration of simulations aimed at exploring droop settings.

5.3.3 Automatic Generation Control

A block diagram of the AGC model employed by PSLTDSim is shown in Figure 5.4. Distributed ACE (DACE) may vary quite a lot depending on control parameters. Simulation settings related to frequency bias, ACE integrating and filtering, and conditional summing are explained in the following sections.

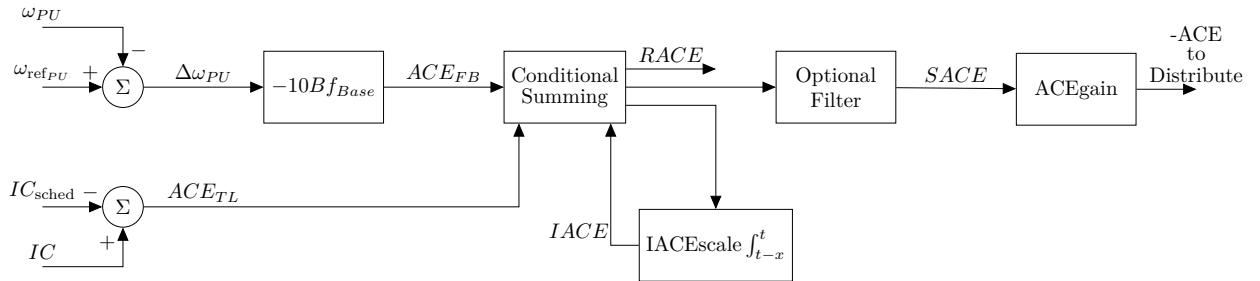


Figure 5.4: Block diagram of ACE calculation and manipulation.

5.3.3.1 Frequency Bias

Choosing a desired frequency bias B can be accomplished in a number of different ways. All methods are configured by the string entered as the ‘B’ value in the sysBA dictionary. The format of the ‘B’ string is ”Float Value : B type”. The available B types are scalebeta, perload, permax, and abs.

The scalebeta type will scale the automatically calculated area frequency response characteristic β by the given float value. The perload type will set B equal to the current area load times the given float value. The permax type will set B equal to the maximum

area capacity times the given float value. The abs type will set B equal to the given float value. Note that the units on B are MW/0.1 Hz and, despite B being a negative number, is entered as a positive value.

A variable frequency bias may be used in distributed ACE signals. If the ‘BVgain’ dictionary entry is a non-zero value, a variable frequency bias B_V is calculated as

$$B_V = B_F (1 + K_B |\Delta\omega|) \quad (5.3)$$

where B_F is the fixed bias value, and K_B is the user entered value for ‘BVgain’. It should be noted that $\Delta\omega$ is calculated according to Equation 2.5 and is a PU value. This leads to seemingly large required values of ‘BVgain’ before effects are noticeable. To clarify, B_V is only used in ACE calculations that are meant to be distributed to the generation fleet, RACE is always calculated using B_F .

5.3.3.2 Integral of Area Control Error

As previously shown in Figure 5.4, ACE may be integrated and fed back into the conditional summing block. The default signal sent to the integrator is RACE as AGC is meant to drive RACE to zero. Settings related to this process are configured in the sysBA dictionary. Integral of ACE (IACE) parameters are ‘IACEwindow’, ‘IACEScale’, and ‘IACEdeadband’. IACEwindow defines the length in seconds of the moving window integrator. If IACEwindow is set to zero, integration will be continuous (i.e. for all time). IACEScale acts as a gain of the output integral value. IACEdeadband specifies the frequency deviation in Hz between which integration values will stop being added into the conditional summing block.

5.3.3.3 Conditional Area Control Error Summing

Depending on the type of AGC agent chosen, ACE is calculated in different ways. The Tie-Line Bias (TLB) agent has multiple types of conditional ACE calculations. All conditionals involve comparing the sign of a value to the sign of frequency deviation. The main idea behind this conditional summing is to ensure that only events occurring inside an

area will receive an AGC response. Options are available to allow for continued frequency response as well. Table 5.1 shows the conditional summations and associated TLB type.

Table 5.1: Tie-line bias AGC type ACE calculations.

TLB Type	ACE Calculation
0	$ACE_{FB} + ACE_{TL}$
1	$ACE_{FB} + ACE_{TL} * [\text{sgn}(\Delta\omega) == \text{sgn}(ACE_{TL})]$
2	$ACE_{FB} + ACE_{TL} * [\text{sgn}(\Delta\omega) == \text{sgn}(ACE_{FB} + ACE_{TL})]$
3	$ACE_{FB} * [\text{sgn}(\Delta\omega) == \text{sgn}(ACE_{FB})] + ACE_{TL} * [\text{sgn}(\Delta\omega) == \text{sgn}(ACE_{TL})]$
4	$[ACE_{FB} + ACE_{TL}] * [\text{sgn}(\Delta\omega) == \text{sgn}(ACE_{FB} + ACE_{TL})]$

5.3.3.4 Area Control Error Filtering

The calculated ACE can be put through a filter, or smoothed, to become smoothed ACE (SACE). The three basic filters created were low pass, integral, and proportional and integral (PI). Block diagrams of these filters are shown in Figure 5.5.

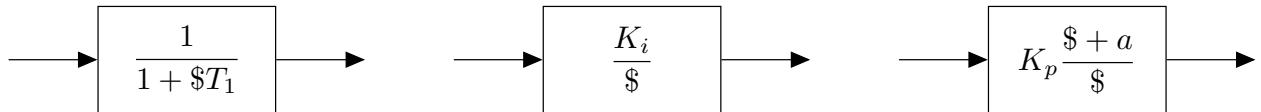


Figure 5.5: Block diagrams of optional ACE filters.

The selection and configuration of a filter is done in the BA parameter dictionary via the ‘ACEFiltering’ key value. The format of the string input to the ‘ACEFiltering’ key is “type : val1 val2”. Valid filter types are lowpass, integrator, and pi. The low pass and integrator take only one value while the PI filter takes two. In the case of the low pass filter, the passed in value is set as the low pass time constant. The value passed in with an integrator filter is simply a gain. The first PI value is used as a proportional gain value K_p and the second value describes the ratio between integral and proportional gain a .

5.3.3.5 Controlled Generators and Participation Factors

Each BA is configured with a list of controlled generators that receive AGC signals. These generators, or power plants, are given a participation factor between 1 and 0. The participation factor dictates the percent of ACE signal each agent receives. A check is done

to ensure each BA has a total participation factor of one, however, if the sum of participation factors is not one, only a warning is issued. It is up to the user to enter reasonable values. Additionally, each list value of the ‘CtrlGens’ key describes if the signal should be applied as a step or a ramp. Ramps are best when single units are receiving ACE, while steps are useful for power plants that are intended to handle distribution of ACE independently.

5.4 Governor Deadband Effect on Valve Travel

Initial research goals included maximizing system stability while minimizing machine effort. The decided upon metric for machine effort was valve travel. As governor deadbands directly affect valve travel, a study into deadbands was conducted using PSLTDSim.

5.4.1 Governor Deadband Simulation Configuration

To assess long-term impacts of governor deadbands, thirty minutes of random load noise was applied to the mini WECC. All governors had identical deadband settings and PSLTDSim was used to set all governor droops to 5%. Some governors were removed from the system so that only $\approx 20\%$ of generation capacity in each area had governor control. Each type of deadband shown in Figure 5.3 was simulated. No-step and non-linear droop deadbands had a threshold of 16 mHz while the step deadband used a 36 mHz deadband. Noise agent N_Z was set to 0.03 for all simulations with random walk behavior enabled. The change in system loading caused by the noise agent is shown in Figure 5.6.

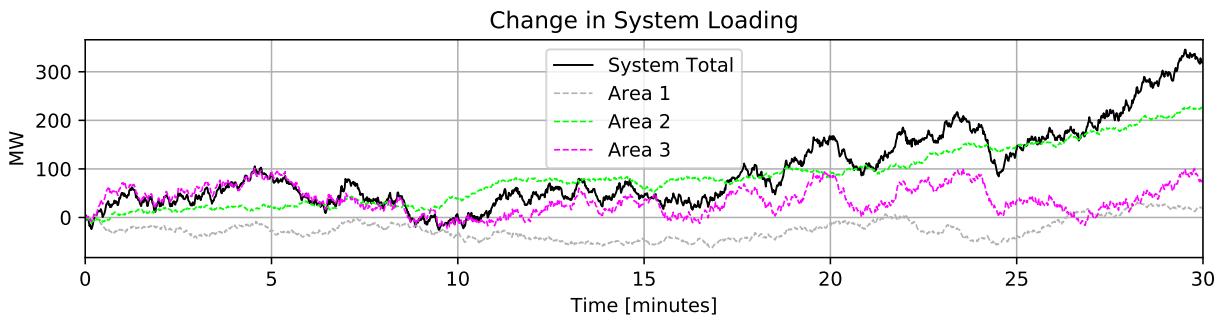


Figure 5.6: Cumulative system change in load for governor deadband simulation.

Another experiment was conducted to explore a non-homogeneous deadband scenario where all deadbands were of the no-step type, but some had different mHz deadbands. Although PSLTDSim can model AGC, it was not enabled for these deadband simulations.

5.4.2 Governor Deadband Simulation Results

Figure 5.7 shows the resulting system frequency for each type of deadband. The step deadband holds frequency almost exactly on the set deadband except when system loading decreases during minutes 7-11. The other deadband options maintain system frequency near their respective mHz setting until loading increases beyond a point near minute 17.

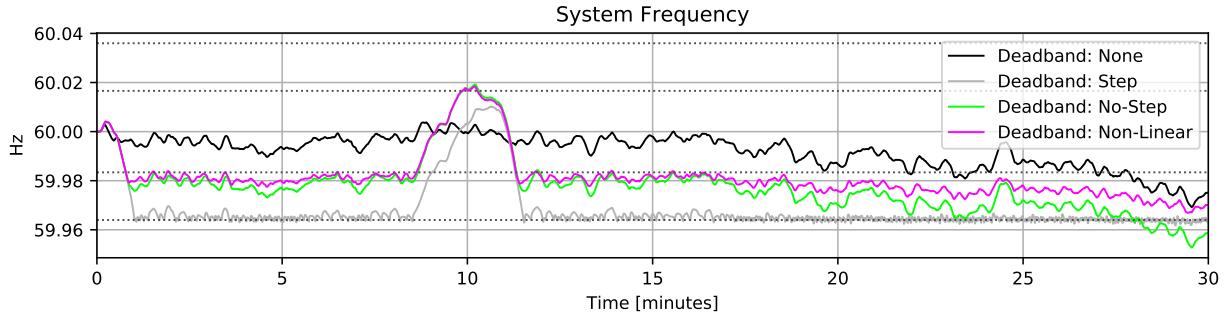


Figure 5.7: System frequency comparison of different deadband scenarios.

The first three minutes of a single generators valve travel are shown in Figure 5.8 to compare how different deadbands affect valve movement. The step type deadband results in pulse train-esq control signals being sent when system frequency is oscillating near the deadband.

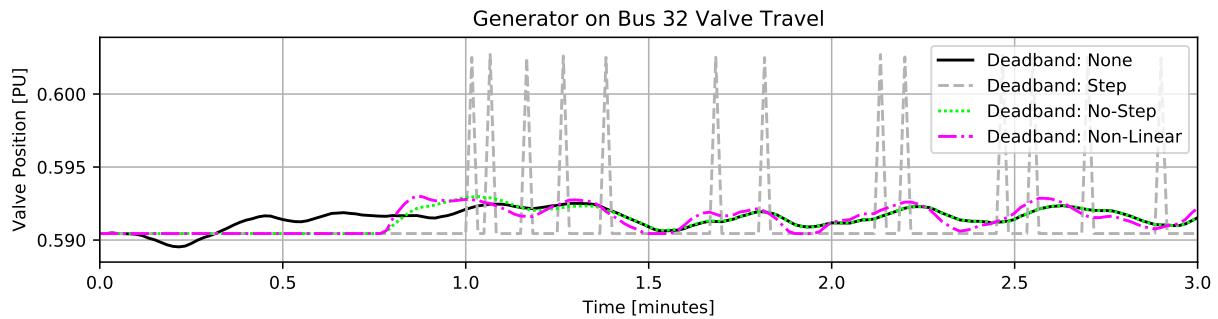


Figure 5.8: Detail comparison of initial valve movement.

With all governors using a no-step type deadband, two of the three areas mHz deadband was set to 16.6 mHz while the third was set to 36 mHz. The left plot of figure 5.9 shows average valve travel over time in a homogeneous deadband system while the right plot shows

non-homogeneous valve travel results. In the homogeneous case, all areas have equal valve travel. In the non-homogeneous case, the larger deadband used in area 3 prevents governor response until minute 18.

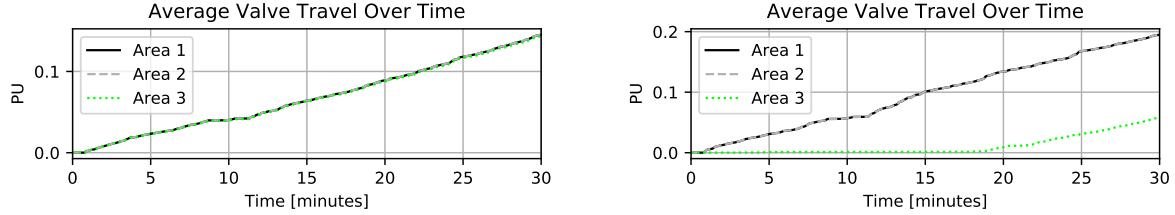


Figure 5.9: Area valve travel over time for homogeneous and non-homogeneous scenarios.

Complete valve travel result plots are presented in Appendix D. A tabular summary of valve travel is shown in Table 5.2. Repeated control pulses associated with a step type deadband greatly increase valve travel over the more linear deadband options. Simulation results reinforce NERC recommendations that step deadbands not be used [18]. When a variety of deadband thresholds are employed, total valve travel may decrease while certain individual movement increases.

Table 5.2: Total valve travel for various deadband scenarios.

Generator	Valve Travel [PU]				
	No DB	Step	No-Step	N-L Droop	No-Step Non-H
17	0.16	7.48	0.15	0.23	0.19
23	0.16	7.48	0.15	0.23	0.19
30	0.16	7.48	0.15	0.23	0.19
32	0.16	7.54	0.15	0.23	0.19
107	0.16	7.54	0.15	0.23	0.19
41	0.15	6.44	0.14	0.23	0.06
45	0.15	6.44	0.14	0.23	0.06
53	0.16	7.54	0.15	0.23	0.06
59	0.15	6.44	0.14	0.23	0.06
Total:	1.41	64.38	1.32	2.07	1.19

5.5 Automatic Generation Control Tuning

After a contingency, AGC acts to restore nominal operating conditions. Long-term simulation is required to simulate AGC action as gentle system recovery takes multiple minutes. According to [31], AGC should respond only to internal events or to correct frequency. PLSTDSim simulations using conditional AGC provide results that show conflicting AGC action extends recovery time and increases machine effort.

5.5.1 AGC Simulation Configuration

Using the two area six machine system, a 150 MW loss of generation event in each area was used to tune AGC response to a large contingency. AGC settings were manipulated until an individual BA could restore system frequency in less than 10 minutes. The effect of noise and non-linear governor deadbands was also simulated. Random noise added to each simulation is shown in Figure 5.10. Conditional ACE was used to show conflicting control effort when a BA responds to out-of-area events.

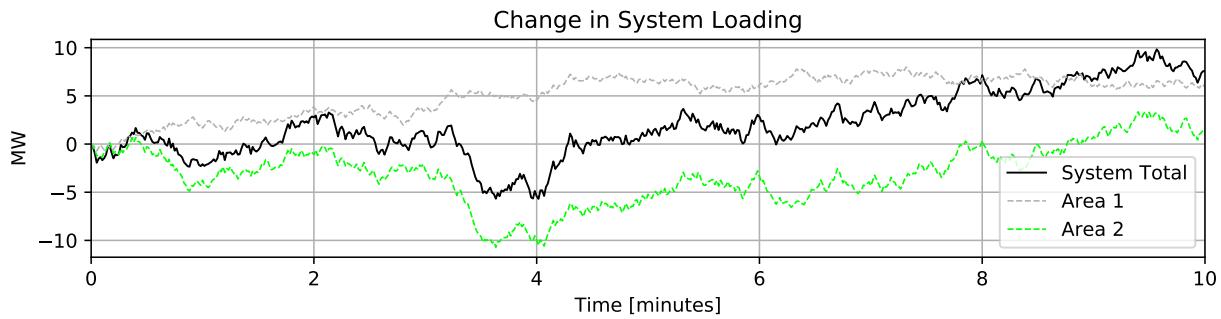


Figure 5.10: Random noise added to AGC simulations.

Full code for simulating an external area event with tuned conditional AGC is shown in Figures B.2 and B.3 in Appendix B.

5.5.2 AGC Simulation Results

5.5.2.1 Base Case Results

The non-AGC frequency response to a -150 MW step in generator power is shown in Figure 5.11. Calculated BA values are shown in Figure 5.12. During the base case, AGC gain is set to zero so no generation changes occur. As area 2 results are largely similar to area 1 results, they are included in Appendix E.

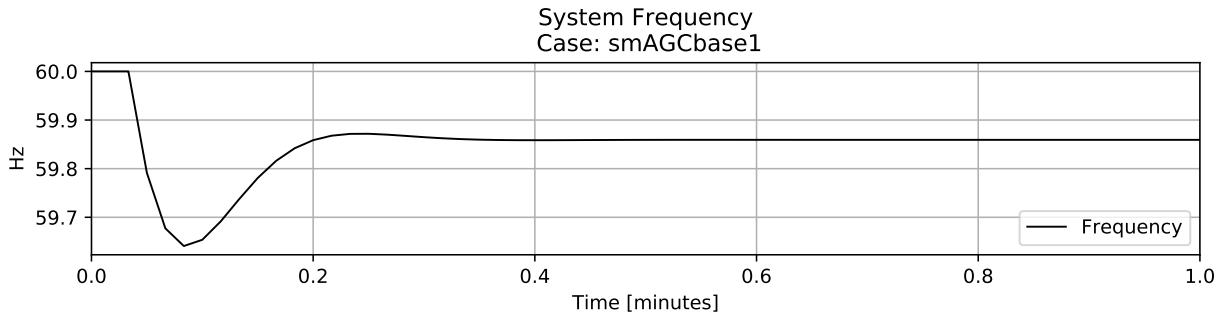


Figure 5.11: Frequency response to generation loss event in area 1.

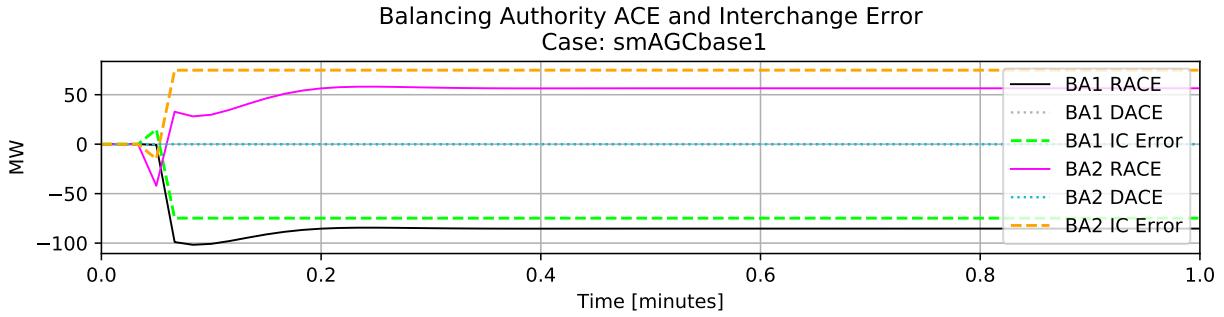


Figure 5.12: Calculated BA values during generation loss event in area 1.

5.5.2.2 AGC Tuning Results

The BA controlling area 1 was equipped with an AGC routine that included a scaled window integrator, PI smoothed ACE, and an action time of 30 seconds. Resulting frequency response is shown in Figure 5.13. Calculated RACE, interchange (IC) error, and distributed ACE (DACE) are shown in Figure 5.12. In a two area system, IC error is symmetric and scaling of RACE by frequency bias can be clearly seen.

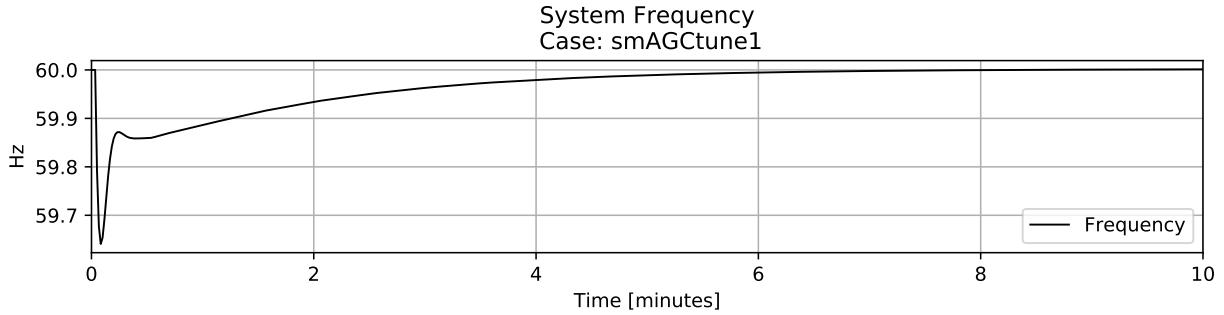


Figure 5.13: AGC Frequency response to area 1 base case scenario.

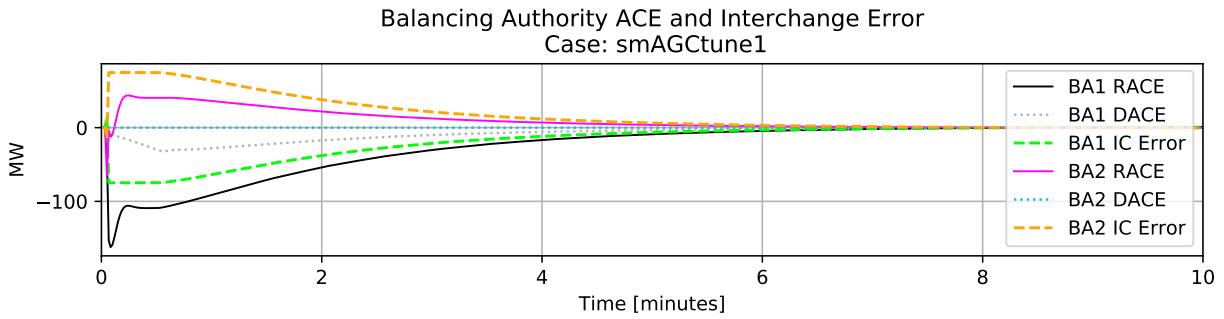


Figure 5.14: Calculated BA values during area 1 AGC tuning.

Figures 5.15 and 5.16 show governor power output and power reference set point responses for each area. During AGC tuning, ACE gain was set to zero for the BA routine not being tuned. As such, area 2 has no P_{ref} changes while area 1 adjusts its controlled governors to return RACE to zero. The control changes generator 1 and generator 2 1 is identical and thus appears as only one set of data.

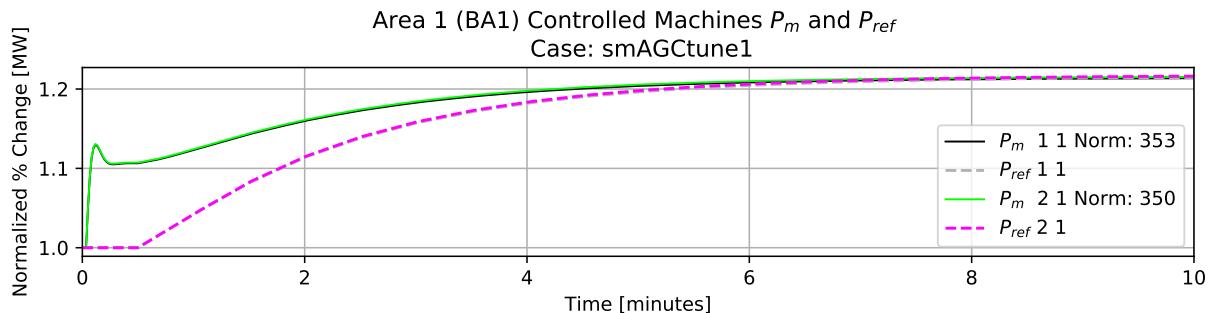


Figure 5.15: Area 1 controlled generation response during AGC tuning.

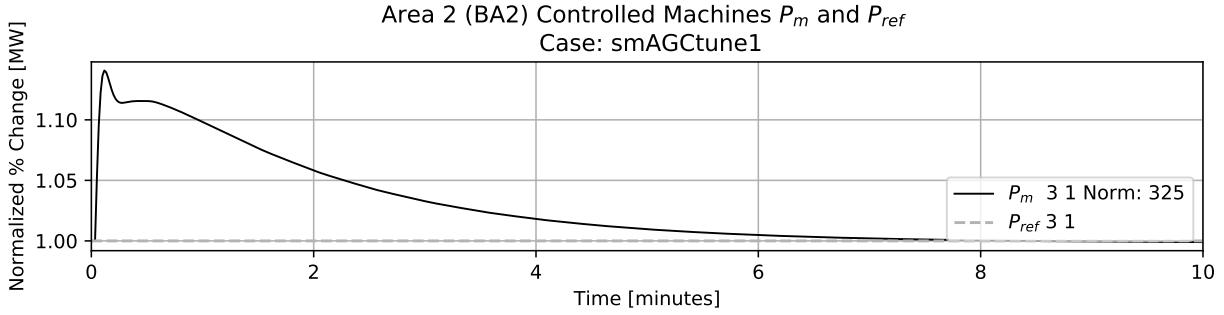


Figure 5.16: Area 2 controlled generation response during AGC tuning.

5.5.2.3 Noise and Deadband Simulation Results

To add slightly more realism to the event, noise and governor deadbands were added to the simulation. Noise was set to 0.05% with random walk enabled. All AGC controlled governors were of the non-linear droop variety with an α of 16 mHz and a β of 36 mHz. Resulting frequency is shown in Figure 5.17. System frequency oscillations between governor deadbands occurs from roughly minute 6 onwards.

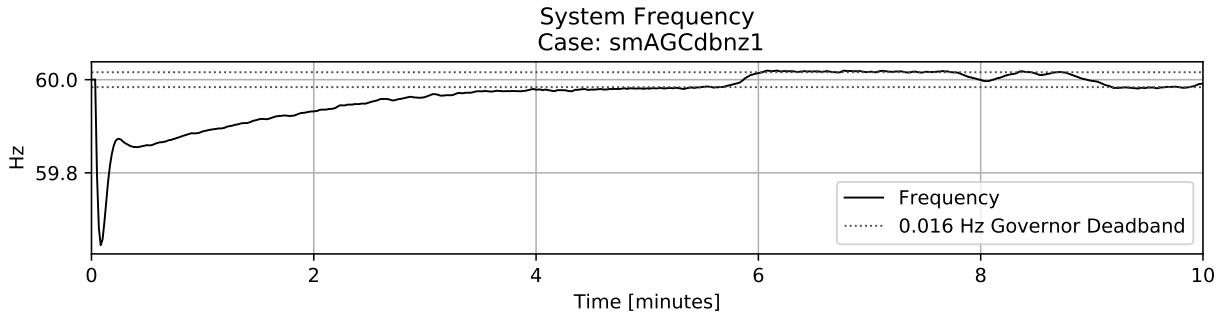


Figure 5.17: AGC frequency response with noise and deadbands.

Figures 5.18, 5.19, and 5.20 show calculated BA values and individual area controlled machine responses respectively. Despite the addition of noise and governor deadbands, system recovery is similar to ideal simulation conditions.

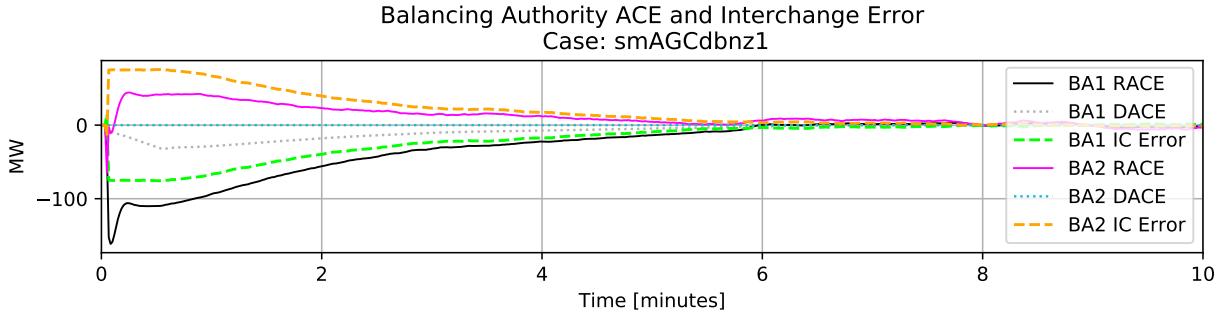


Figure 5.18: Calculated BA values with noise and deadbands.

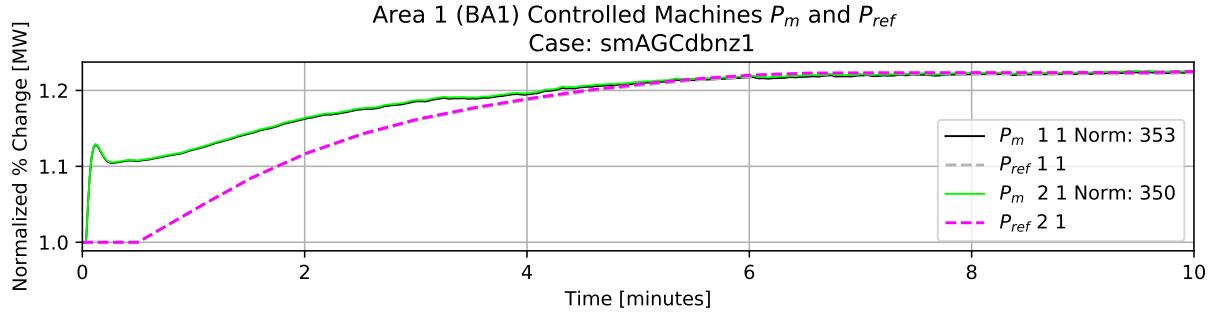


Figure 5.19: Area 1 controlled generation response to noise and deadbands.

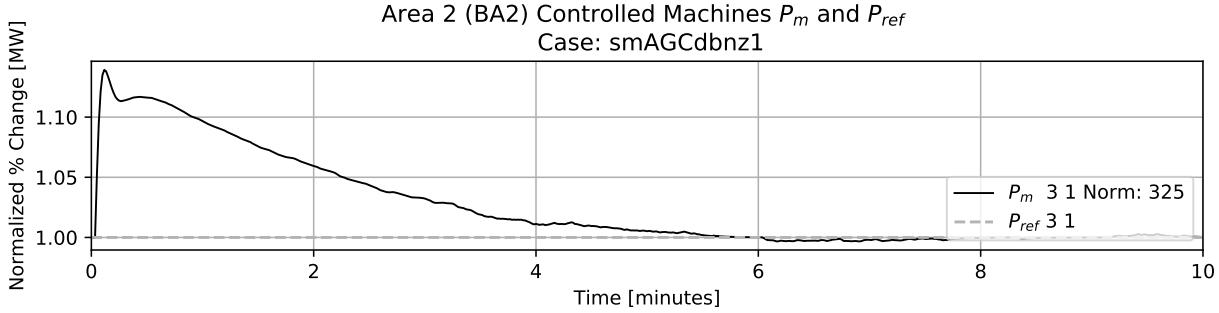


Figure 5.20: Area 2 controlled generation response to noise and deadbands.

5.5.2.4 Conditional ACE Results

After tuning for both BA AGC routines was complete, conditional ACE could be tested. Comparing TLB type 0, which is non-conditional, and conditional TLB type 4 involved enabling AGC action for both areas and simulating an in-area, and out-of-area event. Figure 5.21 shows that system frequency does not return to the nominal operating

point in 10 minutes when TLB 0 is used. Figure 5.22 shows that BA2 DACE acts opposite BA1 DACE.

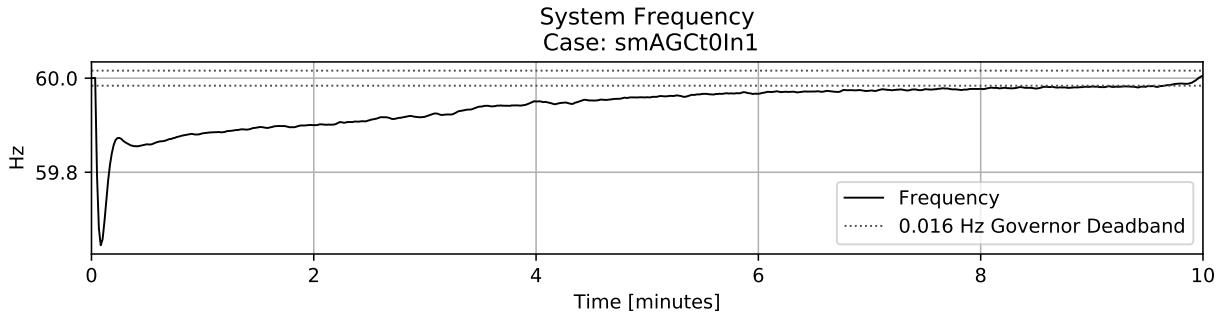


Figure 5.21: Frequency response to event using TLB 0.

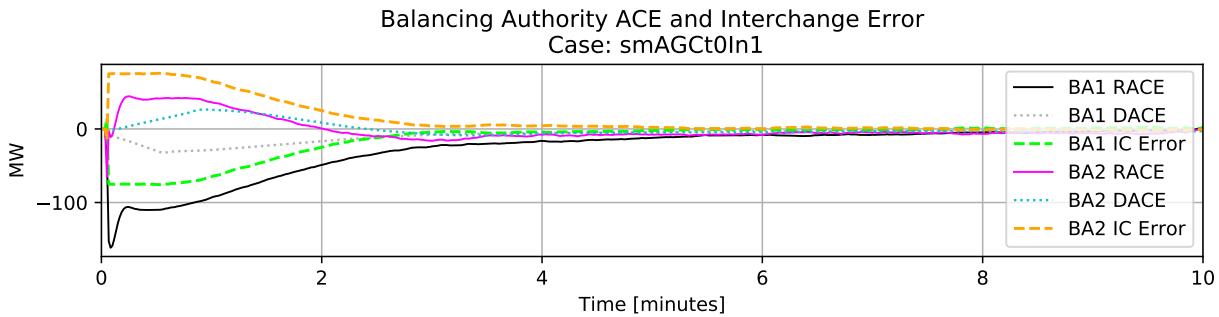


Figure 5.22: Calculated BA values during an event using TLB 0.

To more clearly show conflicting control action, Figures 5.23 and 5.24 show individual area control responses. While BA1 acts to restore frequency by increasing generator output, BA2 acts to restore area interchange by reducing generator output. These control actions are opposite and thus prolong system recovery.

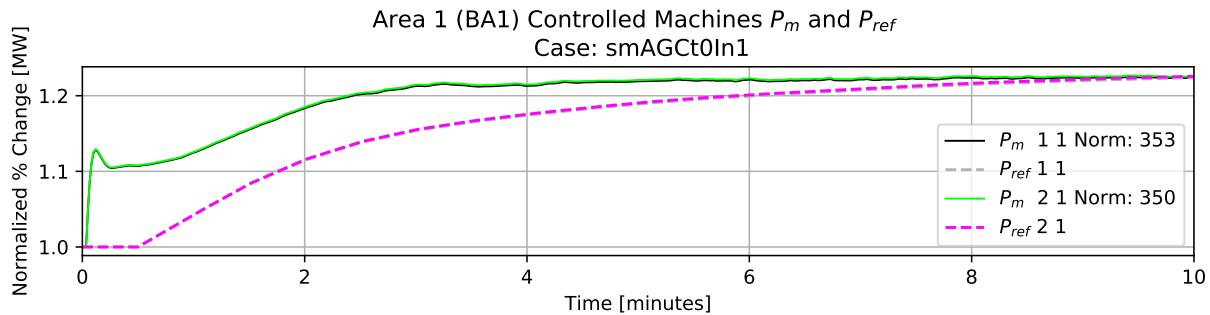


Figure 5.23: Area 1 controlled generation response to internal area event using TLB 0.

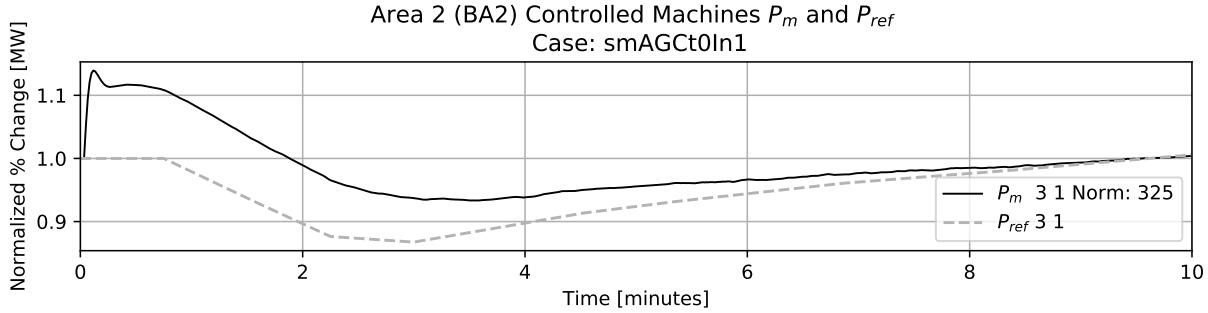


Figure 5.24: Area 2 controlled generation response to external area event using TLB 0.

Figure 5.25 shows system frequency response when each BA send conditional ACE according to TLB type 4 rules. Similar behavior to the noise and deadband only scenario is reproduced.

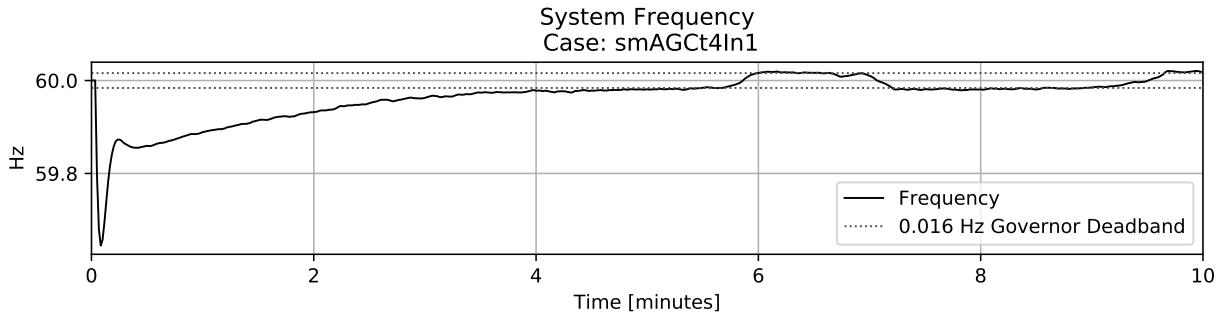


Figure 5.25: Frequency response to event using TLB 4.

Calculated BA values in Figure 5.26 show that BA2 DACE is zero for the event. This is desired as the event is external to area 2 and should not be responded to.

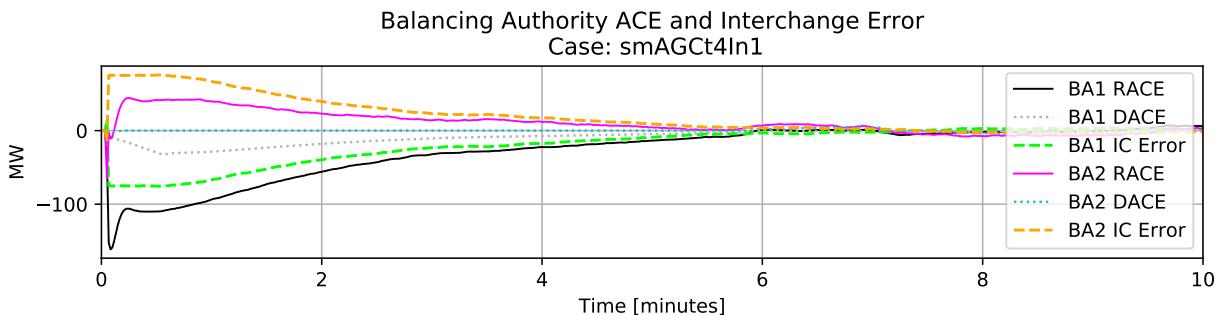


Figure 5.26: Calculated BA values during an event using TLB 4.

Figures 5.27 and 5.28 show controlled machine response is similar to ideal conditions. The P_{ref} AGC response in area 2 near minute 7 is believed to be due to a combination of random load changes affecting area interchange and frequency oscillations caused by governor deadbands.

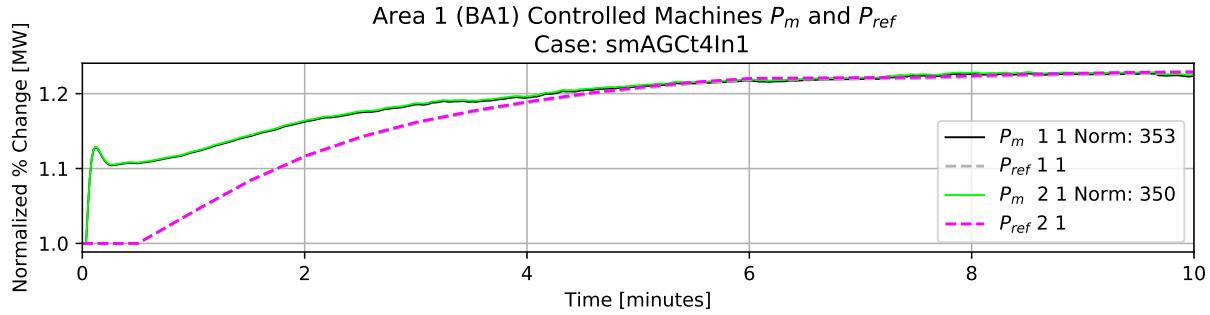


Figure 5.27: Area 1 controlled generation response to internal area event using TLB 4.

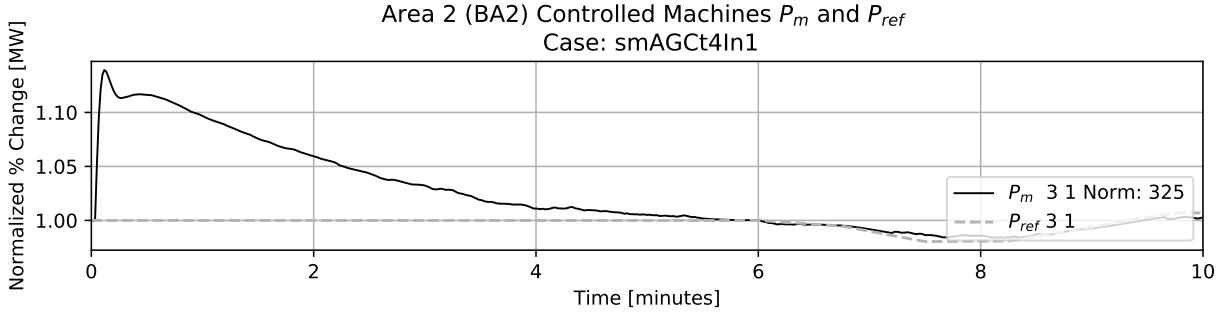


Figure 5.28: Area 1 controlled generation response to external area event using TLB 4.

While an internal event to area 1 is external to area 2, and vice versa, to thoroughly test conditional AGC behavior, an event was simulated in area 2 using both TLB 0 and TLB 4. Results were similar to previously conducted conditional AGC tests. For completeness, plotted results are presented in Appendix E.

In general, simulations show governor deadbands may lead to frequency oscillation between response thresholds and conditional AGC can be used to avoid contradictory recovery action between areas.

5.6 Long-Term Simulation with Shunt Control

Long-term simulations of interest required shunt control to manage voltage. Without voltage control, load changes eventually caused the power-flow solution to diverge. Scenarios chosen for simulation were a four hour morning peak and a two hour virtual wind ramp.

5.6.1 Morning Peak Forecast Demand Simulation

The same six machine two area system and tuned AGC controllers from the previous section were used for long-term simulations. Definite time controllers (DTCs) were used to switch shunts according to bus voltage. Publicly available EIA data was parsed and used to parameterize hourly forecast and demand agents. Data was selected from the morning peak on December 11, 2019 starting at 5:00 AM as reported by the Bonneville Power Administration and California ISO BAs. Normalized BA area power changes are shown in Figure 5.29. Simulated BA1 followed Bonneville data while BA2 adhered to California data. The selected forecast scenario was simulated under ideal conditions and then with the inclusion governor deadbands and load noise. Figure 5.30 shows the random noise added to area 2 causes load value to decrease much more than area 1. A .ltd.py file for the forecast demand scenario with noise and deadbands is shown in Appendix B as Figure B.4.

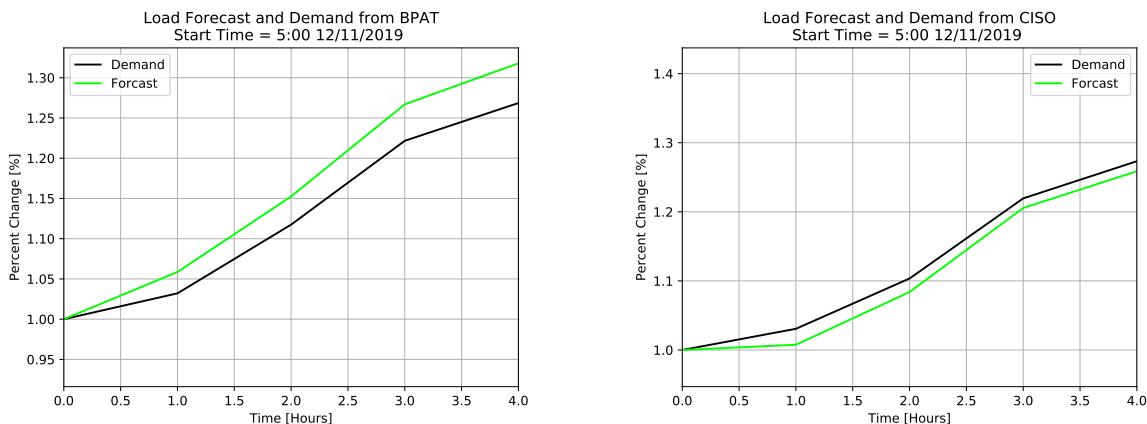


Figure 5.29: Normalized forecast and demand of parsed EIA data.

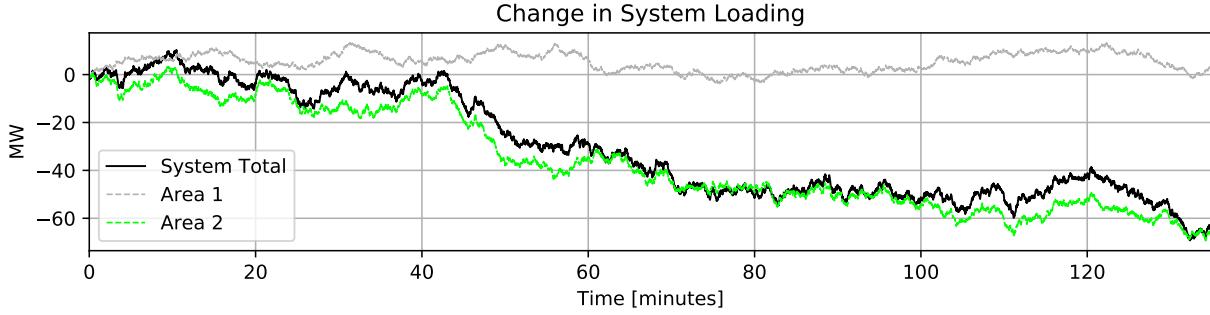


Figure 5.30: Changes in load caused by noise agent action during virtual wind ramp.

5.6.1.1 Morning Peak Forecast Demand Results

Figures 5.31 and 5.32 show the gradual increase of area real power load P , and generated electrical power P_e . Near constant differences between generation and load is maintained by AGC action throughout the entire simulation.

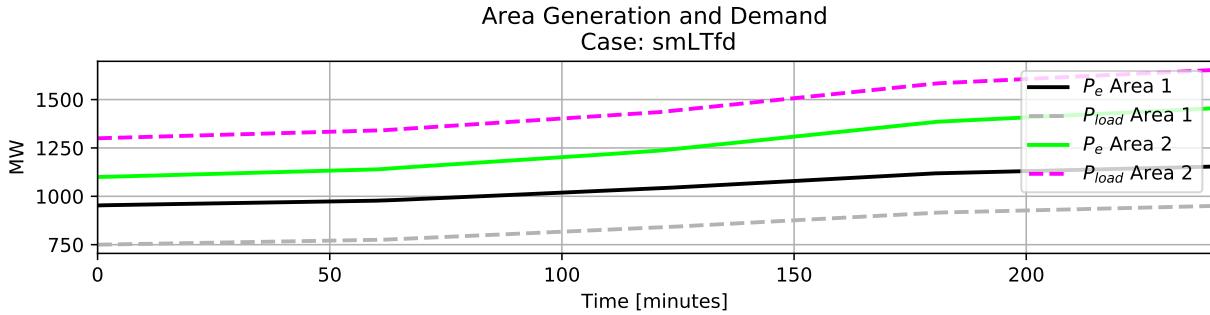


Figure 5.31: Area P_e and Load during morning peak simulation.

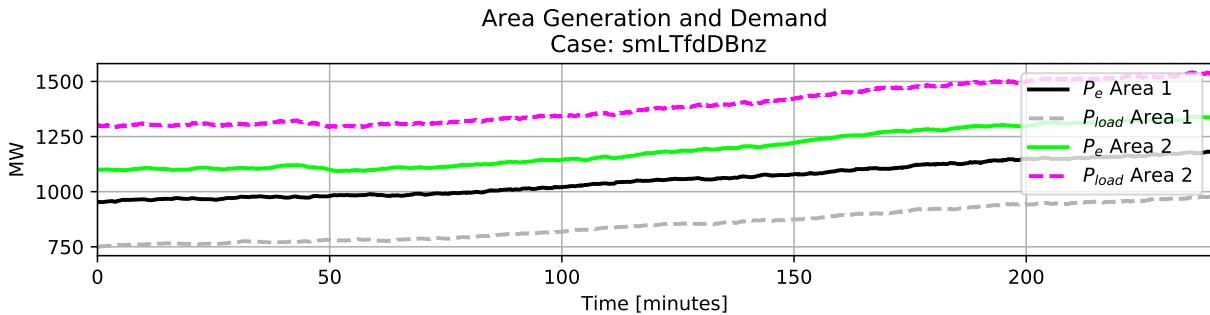


Figure 5.32: Area P_e and Load during morning peak simulation with load noise and governor deadbands.

AGC action also maintains system frequency near the nominal values. Figure 5.33 shows that system frequency response without the addition of deadbands or noise varies less than 1.5 mHz. Figure 5.34 shows that when deadbands and noise are included, frequency tends to oscillate between governor deadbands at a rate of approximately 4 mHz.

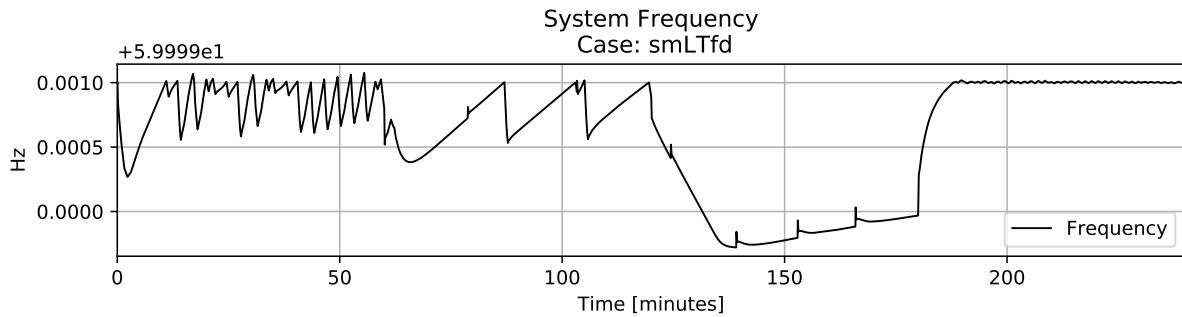


Figure 5.33: System Frequency during morning peak simulation.

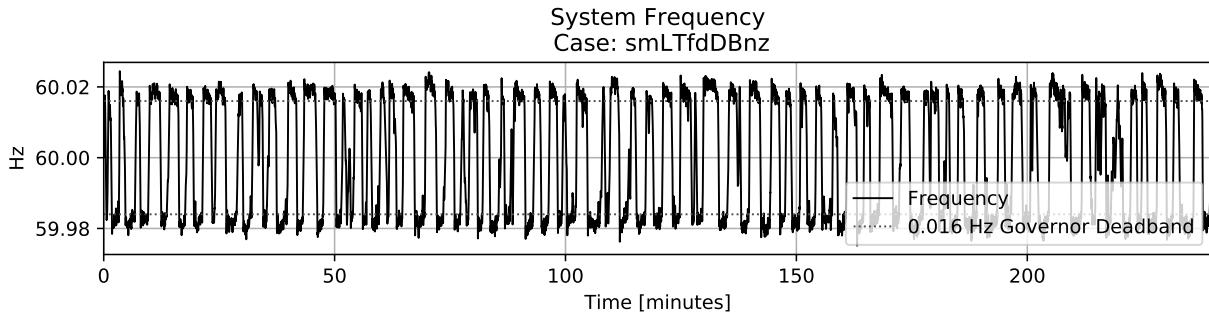


Figure 5.34: System Frequency during morning peak simulation with load noise and governor deadbands.

Figure 5.35 shows calculated BA values during an ideal scenario never exceed a magnitude of 1 MW. Calculated values when noise is included become more busy, but generally stay between ± 10 MW.

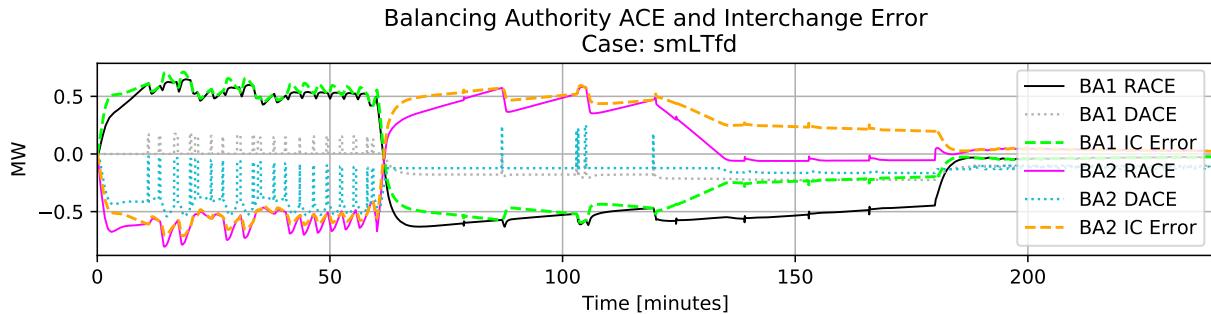


Figure 5.35: Calculated BA values during morning peak simulation.

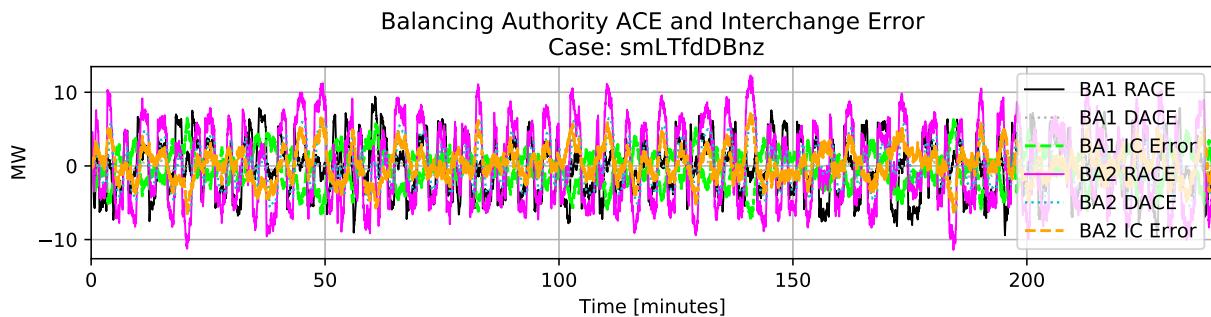


Figure 5.36: Calculated BA values during morning peak simulation with load noise and governor deadbands.

While AGC handles the balancing of generation to load, and thus frequency, voltage must be handled via DTC action. Figures 5.37 and 5.38 show shunt bus voltage under ideal conditions and with load noise and governor deadbands, respectively. As load increases, bus voltage drops. The steps in voltage are caused by capacitive shunts being switched in according to programmed DTC parameters. The random noise added generally reduces load, so shunt switching is slightly delayed when noise is applied compared to the ideal scenario.

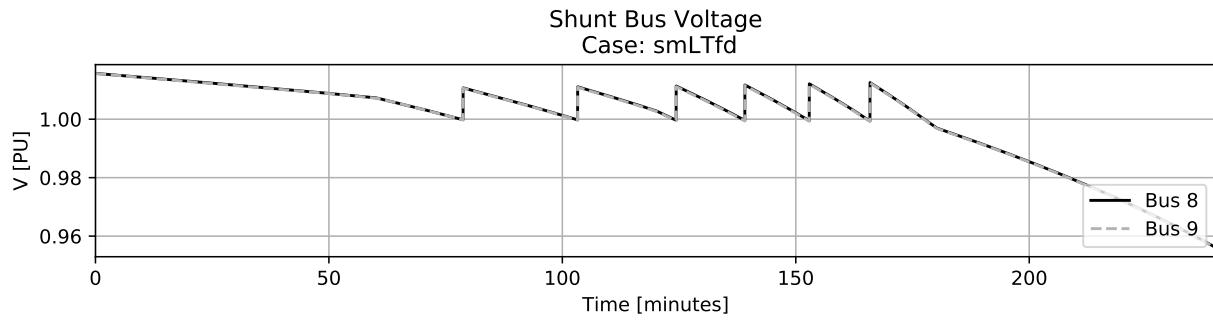


Figure 5.37: System shunt bus voltage during morning peak simulation.

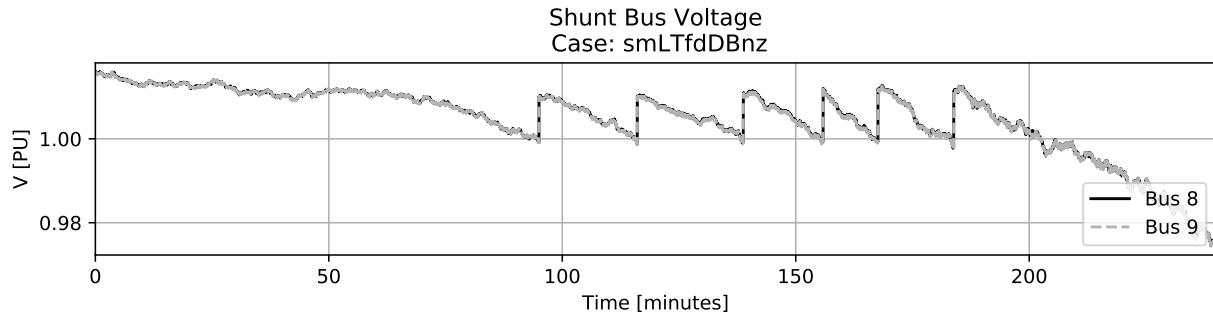


Figure 5.38: System shunt bus voltage during morning peak simulation with load noise and governor deadbands.

Figures 5.39 and 5.40 show the active capacitive load on system bus 8 and 9. DTC action switches capacitors on as voltage drops according to user input logic. Bus 8 has faster acting shunts and thus switches all available caps on before any on bus 9 can respond. Again, the random noise causes a delay in voltage drop, and thus shunt switching as the random noise acts to mostly reduce load.

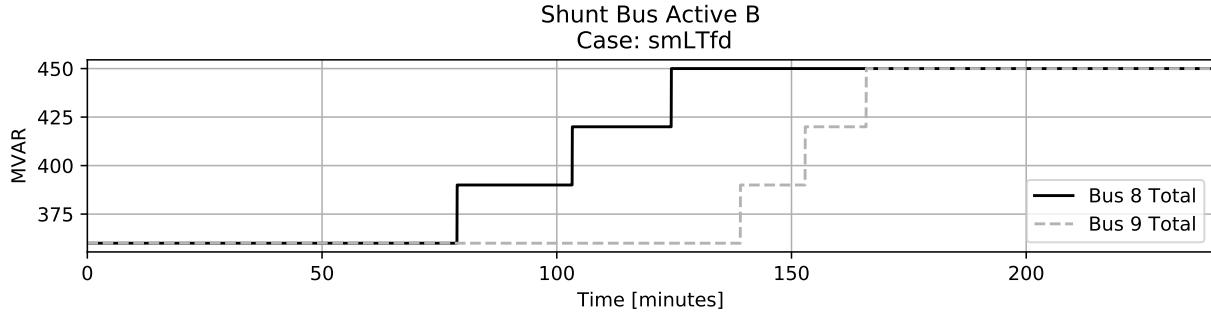


Figure 5.39: System shunt bus MVAR during morning peak simulation.

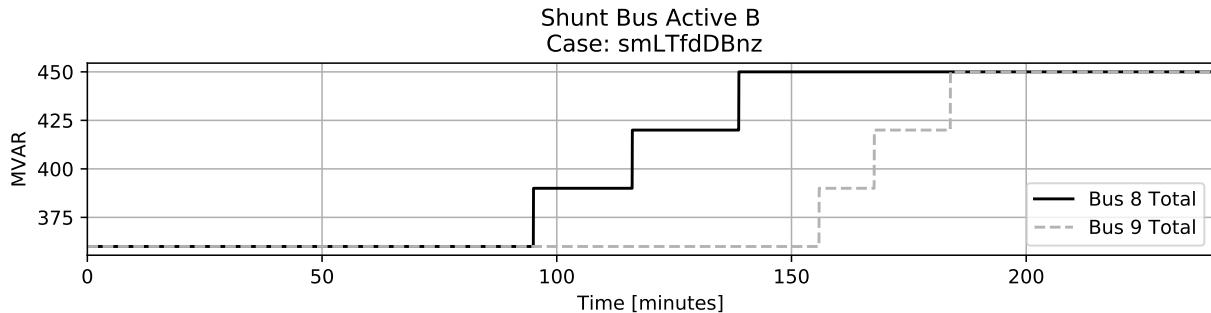


Figure 5.40: System shunt bus MVAR during morning peak simulation with load noise and governor deadbands.

5.6.2 Virtual Wind Ramp Simulation

A virtual wind ramp similar to the one created in [29] was simulated using PSLTDSim. Two ungoverned generators were ramped up, held, and then ramped down. More specifically, using the six machine system, generator 2 2 in area 1 was altered 150 MW while generator 5 in area 2 was changed 300 MW. The generation ramps were 45 minute in duration. The first ramp began at $t = 300$, or 5 minutes into the simulation. A pause of 20 minutes is included between the end of the ramp up and beginning of ramp down. The wind ramp is finished by minute 115, but the simulation continues for another 20 minutes so system recovery can be observed. AGC settings for area 2 were modified from the forecast demand case to include generator 4 receiving 30.0% of AGC signals. This was required as AGC would force generator 3 to supply 0 MW otherwise. Random noise injections into the system loads are shown in Figure 5.41.

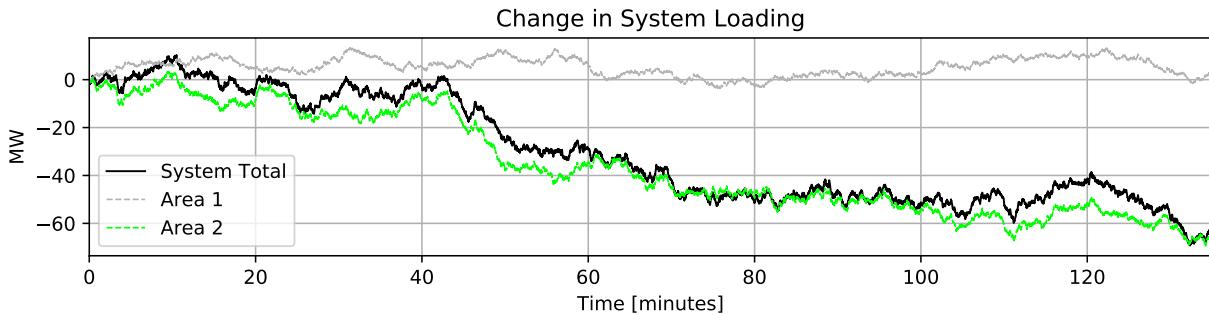


Figure 5.41: Changes in load caused by noise agent action during virtual wind ramp.

5.6.2.1 Virtual Wind Ramp Results

Figures 5.42 and 5.43 show the ramping up behavior of generators 2 2 and 5 as well as the AGC effect on controlled generators. General behavior observed with added load noise and governor deadbands is similar to the ideal case.

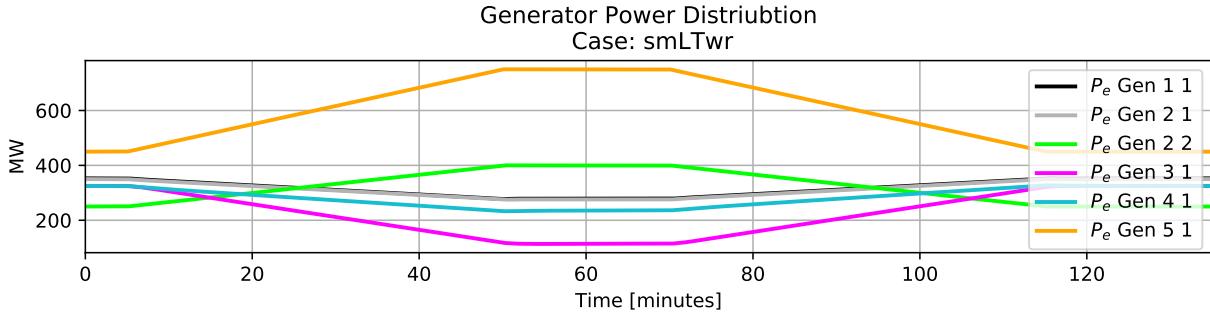


Figure 5.42: Generator P_e distribution during virtual wind ramp under ideal conditions.

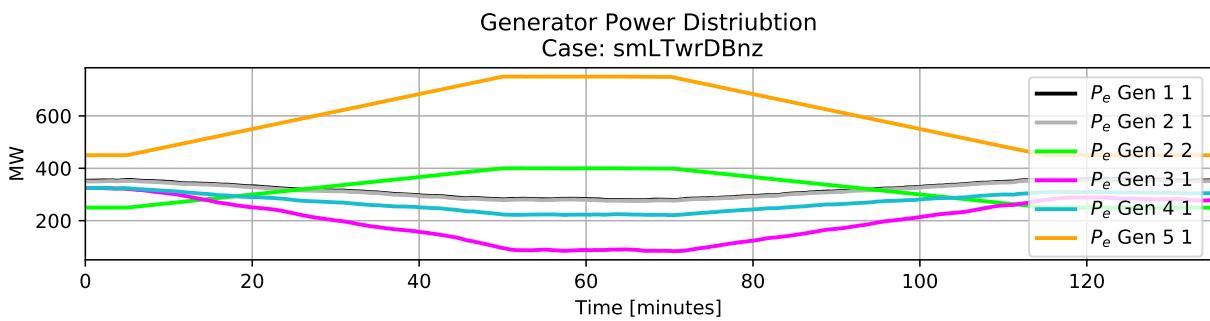


Figure 5.43: Generator P_e distribution during virtual wind ramp with load noise and governor deadbands.

Figures 5.44 and 5.45 show system frequency response to a wind ramp under ideal conditions and with governor deadbands and load noise included respectively. In both cases, AGC is not able to achieve nominal system frequency during ramp events. Because the ramps are 45 minutes, this maintained frequency deviation is in violation of interpreted NERC mandate BAL-002-3. When noise and deadbands are included, system frequency oscillates around deadband thresholds between ramp events.

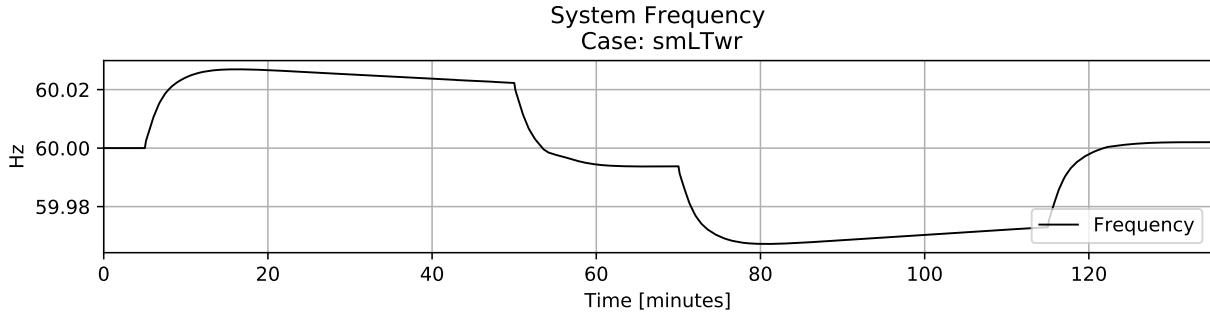


Figure 5.44: System frequency response to virtual wind ramp under ideal conditions.

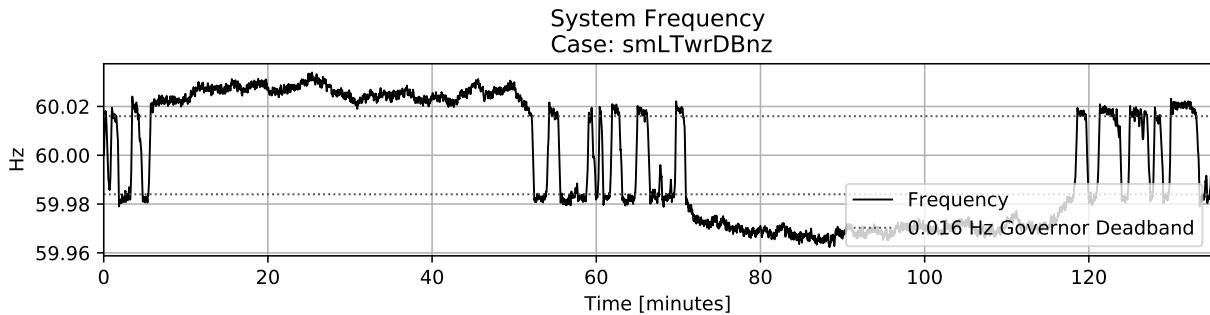


Figure 5.45: System frequency response to virtual wind ramp with load noise and governor deadbands.

Figure 5.46 shows calculated BA values during the virtual wind ramp under ideal conditions. It can be seen that DACE does not match RACE values during ramp events and that IC error is symmetric. Figure 5.47 shows calculated BA with noise and deadbands are slightly larger in magnitude. In both either case, maximum RACE does not exceed ± 15 MW.

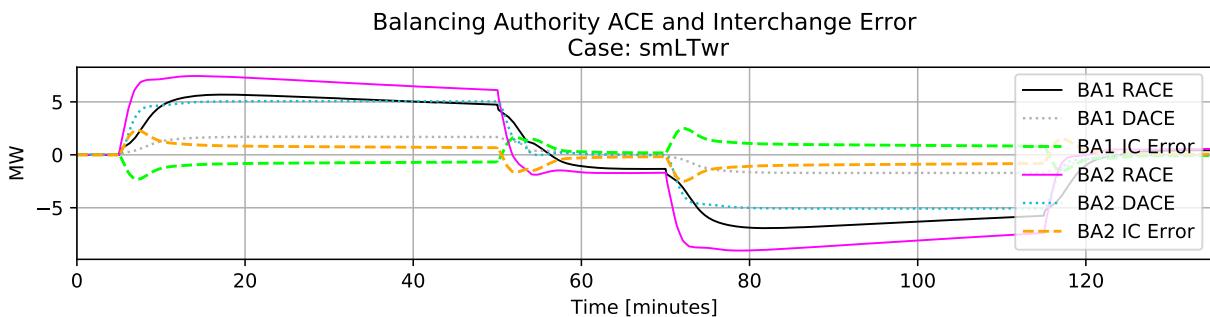


Figure 5.46: Calculated BA values during virtual wind ramp under ideal conditions.

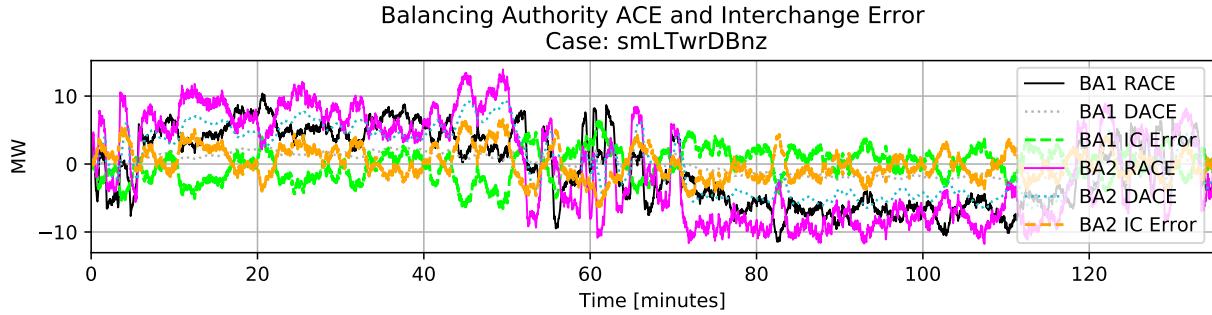


Figure 5.47: Calculated BA values during virtual wind ramp with load noise and governor deadbands.

Figure 5.48 shows that AGC does a good job of maintaining near constant generator output P_e and that load does not change during the ideal scenario. Figure 5.49 shows similar AGC action despite added noise decreasing area 2 loading.

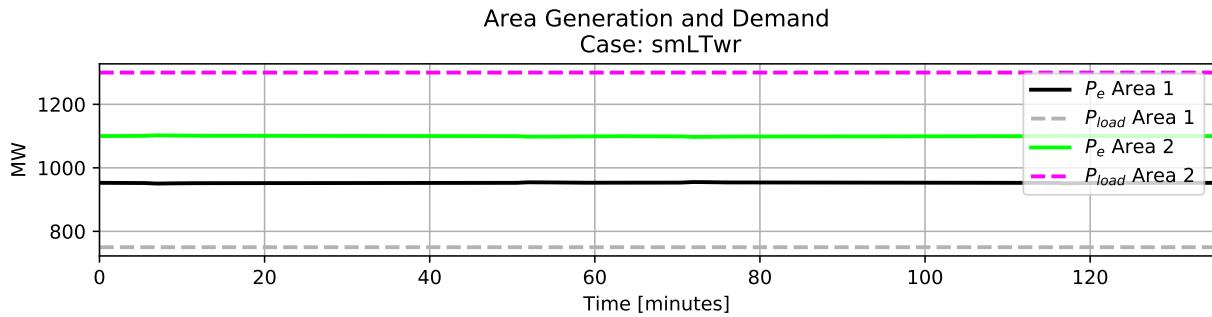


Figure 5.48: Area P_e and P_{load} during virtual wind ramp under ideal conditions.

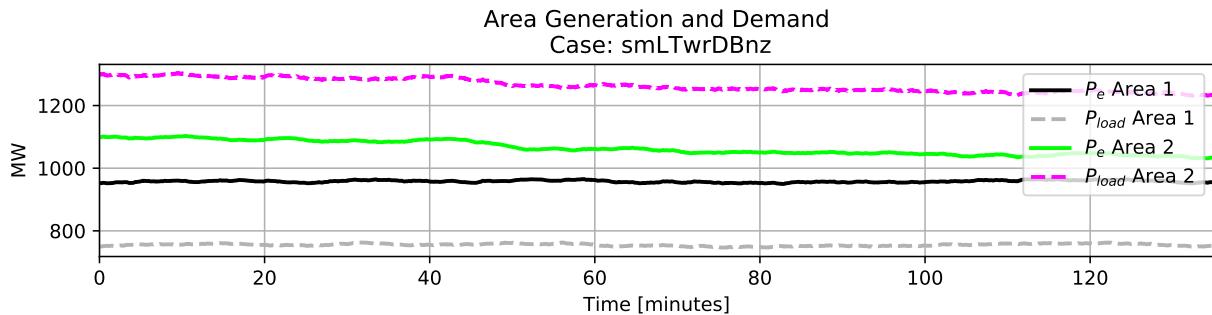


Figure 5.49: Area P_e and P_{load} during virtual wind ramp with load noise and governor deadbands.

Figures 5.50 and 5.51 show that bus voltage between the two scenarios was fairly

similar. As the wind ramps up, bus voltage drops and switched shunts step on to increase voltage. When the wind ramps down, the opposite is true. The decreased load caused by random noise triggers an additional capacitor to be switched off near minute 115.

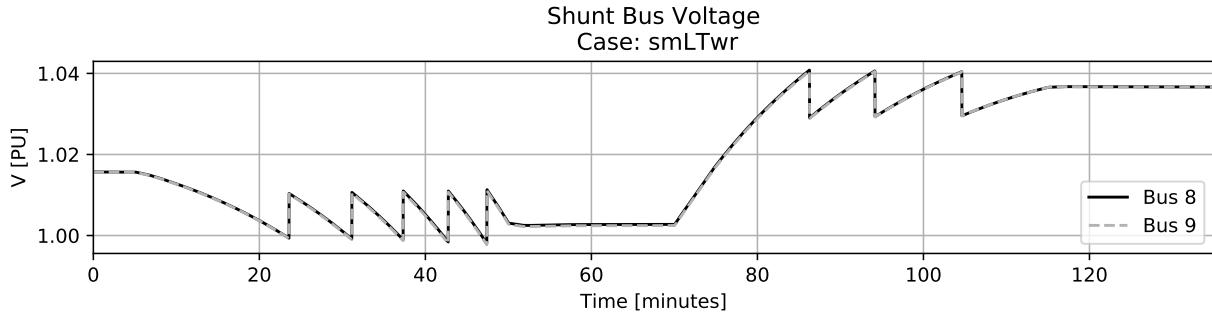


Figure 5.50: System shunt bus voltage during virtual wind ramp under ideal conditions.

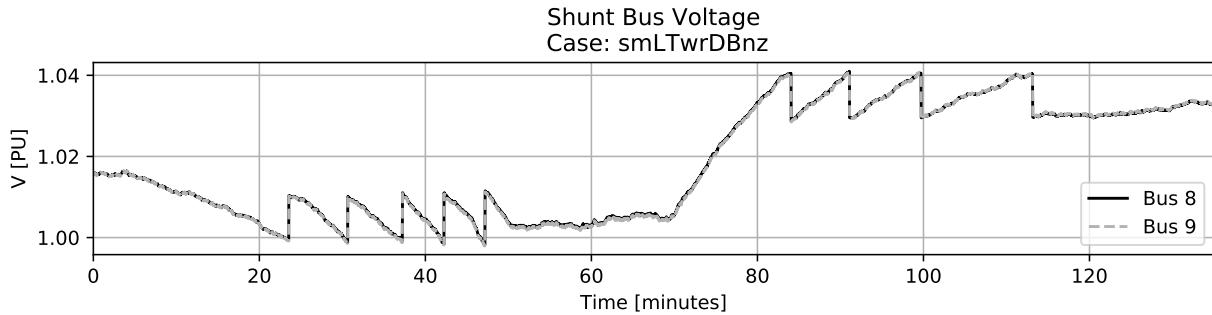


Figure 5.51: System shunt bus voltage during virtual wind ramp with load noise and governor deadbands.

Figure 5.52 shows the MVAR injections caused by switched shunts under ideal conditions. As bus 8 has a ‘faster’ DTC routine, all available capacitors are turned on before bus 9 shunts activate during the ramp up event. The same is true when the wind ramps down; which leaves bus 9 shunts on. Figure 5.53 shows that added noise creates a lower voltage on bus 9 and enables a bus 9 shunt to switch on before all bus 8 capacitors are deployed. The additional bus 8 cap removal near time 115 is also caused by random noise, but is obscured by the plot legend.

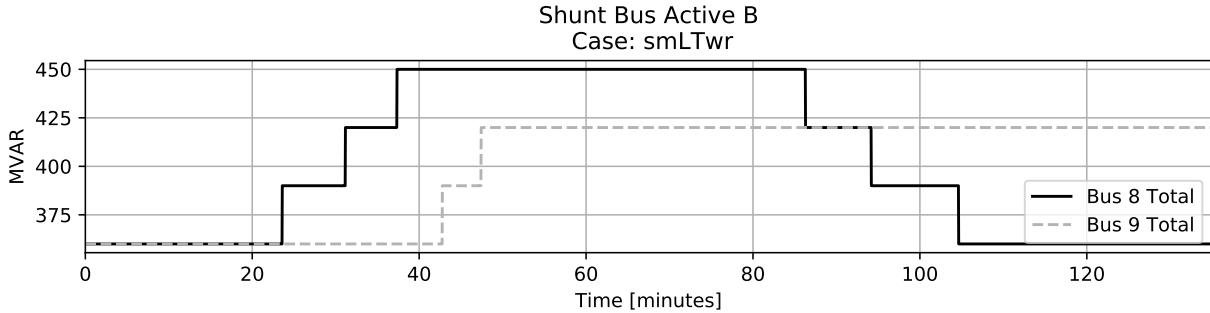


Figure 5.52: System shunt bus MVAR during virtual wind ramp under ideal conditions.

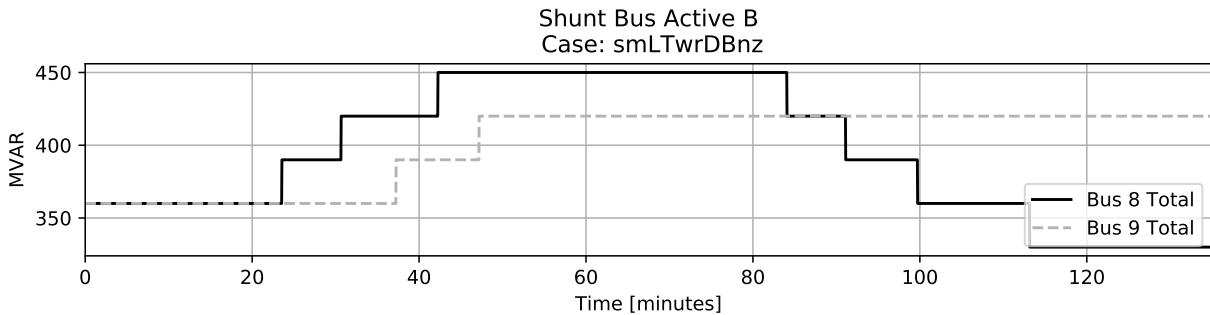


Figure 5.53: System shunt bus MVAR during virtual wind ramp with load noise and governor deadbands.

5.6.3 Long-Term Simulation with Shunt Control Result Summary

In general, configured AGC handles area interchange well, but may require further tuning to adequately handle frequency during long ramps. Simulated governor deadbands introduce system frequency oscillation between deadband thresholds. Bus voltage reference signals can be used to manage switched shunt operation. Addition of random noise may cause very slight differences that have noticeable effects on automatic system response.

Questionable AGC results in the ideal wind ramp case may indicate issues with calculated value behavior. Specifically, DACE never meets a relatively steady RACE value. The inclusion of an IACE term should enable DACE to eventually be larger than RACE.

5.7 Feed-Forward Governor Action

PSLTDSim can be used to recreate system behavior. Interested parties expressed a concern with an undesirable governor response believed to be caused by feed-forward characteristics. The provided model of governors in question was described as a single block with no further information. A plot of simulated and recorded generator power output was provided as an example of noticed behavior. A similar governor response was created using a DTC and governor input filtering.

5.7.1 Feed-Forward Governor Simulation Configuration

Using a six machine system with 0.5 second time step, a method of stepping governor P_{ref} while also gaining input $\Delta\omega_{PU}$ produced a similar undesirable response. Code used to define system generation step, governor delay, and DTC action is shown in Figure 5.54. In practice, this code would be user defined in the simulation .ltd.py file.

The perturbation list in Figure 5.54 specifies an ungoverned generator on bus 5 with a mechanical power step down of 100 MW at t=20. This was meant to simulate the tripping of a generator in an aggregated generator model. The governor delay dictionary was used to gain the $\Delta\omega$ input by 0.5. This was required so that steady state frequency did not over account for differences of $\Delta\omega$. The DTC action was defined to occur every 24 seconds and sets $P_{ref} = P_{ref0} + \frac{\Delta\omega}{R}M_{base} * 0.5$. While not truly a feed-forward control response, the stepping of P_{ref} was thought to produce a similar result. Action time of 24 seconds was chosen so the first DTC response is near the simulated frequency nadir.

```

1  # Perturbances
2  mirror.sysPerturbances = [
3      'gen 5 : step Pm 20 -100 rel', # Step no-gov generator down
4  ]
5
6  # Delay block used as delta_w gain
7  mirror.govDelay ={
8      'delaygen2' : {
9          'genBus' : 2,
10         'genId' : '1', # optional
11         'wDelay' : (0, 0, .5), # gain of input w
12         'PrefDelay' : (0, 0)
13     },
14     #end of defined governor delays
15 }
16
17 # Definite Time Controller Definitions
18 mirror.DTCdict = {
19     'ffGovTest' : {
20         'RefAgents' : {
21             'ra1' : 'mirror : f',
22             'ra2' : 'gen 2 1 : R',
23             'ra3' : 'gen 2 1 : Pref0',
24             'ra4' : 'gen 2 1 : Mbase',
25         },# end Referenc Agents
26         'TarAgents' : {
27             'tar1' : 'gen 2 1 : Pref',
28         }, # end Target Agents
29         'Timers' : {
30             'set' :{ # set Pref
31                 'logic' : "(ra1 > 0)", # should always eval as true
32                 'actTime' : 24, # seconds of true logic before act
33                 'act' : "tar1 = ra3 + (1-ra1)/(ra2) * ra4 * 0.5 ", # step Pref
34         },# end set
35             'reset' :{ # not used in example
36                 'logic' : "0",
37                 'actTime' : 0, # seconds of true logic before act
38                 'act' : "0", # set any target On target = 0
39         },# end reset
40             'hold' : 0, # minimum time between actions (not used in example)
41         }, # end timers
42     },# end bpaTest
43 }# end DTCdict

```

Figure 5.54: Long-term dynamic settings for feed-forward governor simulation.

5.7.2 Feed-Forward Governor Simulation Results

Simulation results for frequency, and generator electrical power are shown in Figures 5.55 and 5.56 respectively. The undesired response can be seen when power output is stepped beyond the steady state value.

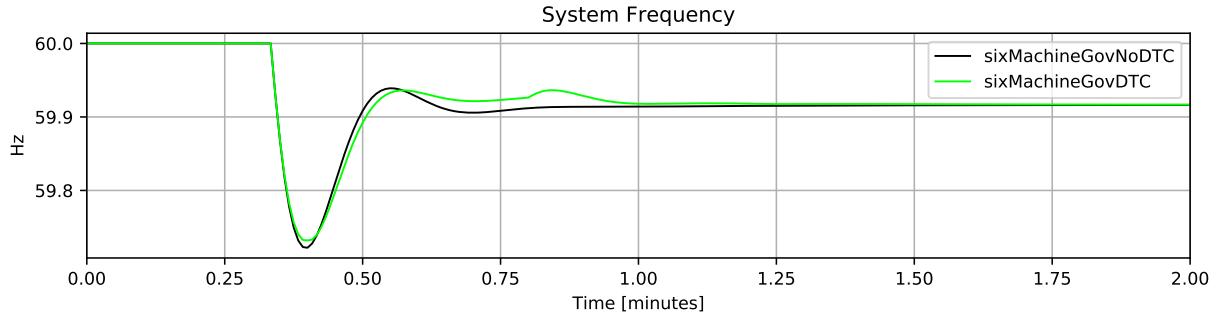


Figure 5.55: Feed-forward governor frequency comparison.

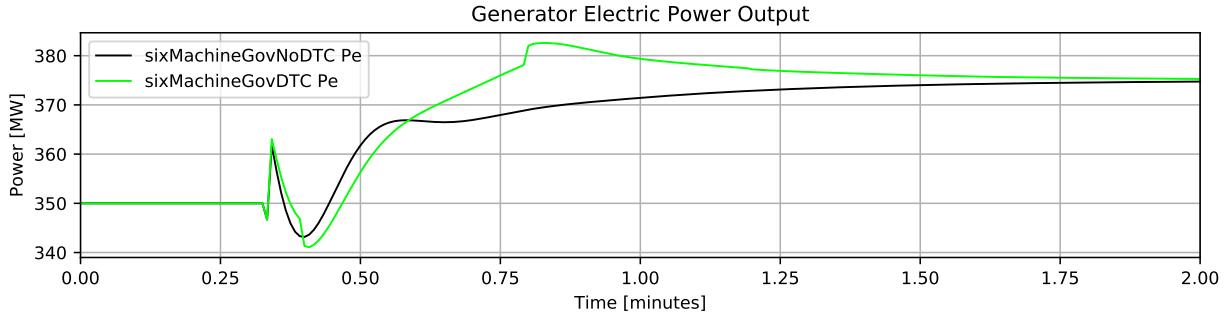


Figure 5.56: Feed-forward governor electric power comparison.

While the produced behavior does resemble observed behavior, it is not an exact match. Scaling and actual response time differences are believed to be caused by differences in model size, governor time constants, and actual feed-forward action. A new governor model with more defined feed-forward capabilities could be created if simulated results on actual system-under-study are unsatisfactory.

5.8 Variable System Damping and Inertia

PSLTDSim can be used to estimate system behavior. To adjust frequency response, system damping and inertia may be manipulated. The trend of increased photo-voltaic penetration can lead to situations where total system inertia may not match what is expected. The single combined system inertia in PSLTDSim can be modified during simulations to enable study into variable inertia system responses.

5.8.1 Damping and Inertia Simulation Configuration

Damping effects were explored by modifying the "Dsys" value in the simParams dictionary of a simulation .py file. System inertia effects were studied by altering the "Hinput" in simParams dictionary and by perturbance agent action. A pulse train of load steps was created to show the ability varying system inertia during a simulation. System inertia was altered before each positive load step. Code used to define perturbance steps is shown in Figure 5.57. In practice, this code would be user defined in the simulation .ltd.py file.

```

1 # Perturbances
2 mirror.sysPerturbances = [
3     # Initial system response
4     'load 8 : step P 2 100 rel',
5     'load 8 : step P 22 -100 rel',
6     # decrease of H
7     'mirror : step Hsys 40 -30 per',
8     'load 8 : step P 42 100 rel',
9     'load 8 : step P 62 -100 rel',
10    # decrease of H
11    'mirror : step Hsys 80 -50 per',
12    'load 8 : step P 82 100 rel',
13    'load 8 : step P 102 -100 rel',
14    # Increase of H
15    'mirror : step Hsys 120 30080 abs',
16    'load 8 : step P 122 100 rel',
17    'load 8 : step P 142 -100 rel',
18]

```

Figure 5.57: Long-term dynamic settings for variable system inertia simulation.

The code in Figure 5.57 steps the load on bus 8 up and down 100 MW every 20 seconds starting at $t=2$. System inertia is reduced by 30% at $t=40$, 50% at $t=80$, and then set to 30,080 MW s at $t=120$.

5.8.2 Damping and Inertia Simulation Results

System damping effects to a 100 MW generator step are shown in Figure 5.58. A positive damping value increases system oscillation while a negative damping value decreases oscillations. Altering system damping affects steady state frequency response.

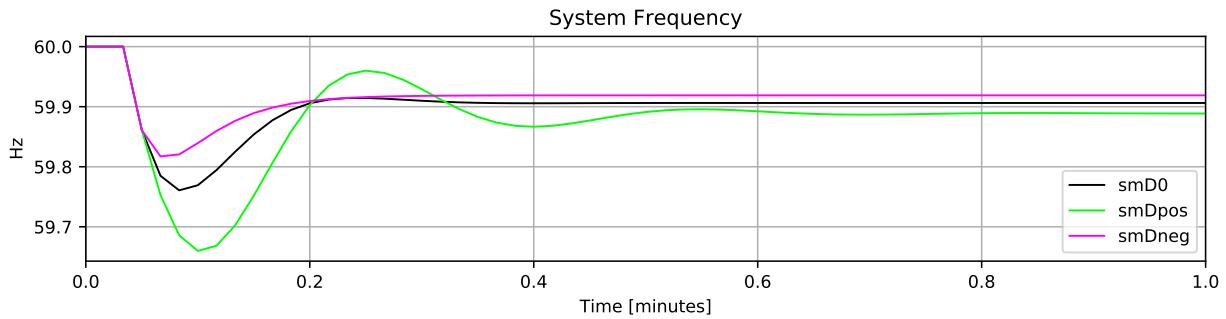


Figure 5.58: Frequency effects of system damping.

Frequency response to varying scalings of system inertia are shown in Figure 5.59. The smH100 case is an un-modified system inertia, while the 90, 80 and 70 case appendings reflect a 90.0%, 80.0%, and 70.0% inertia scaling respectively. A scaling of 60.0% resulted with a power-flow problem that did not converge.

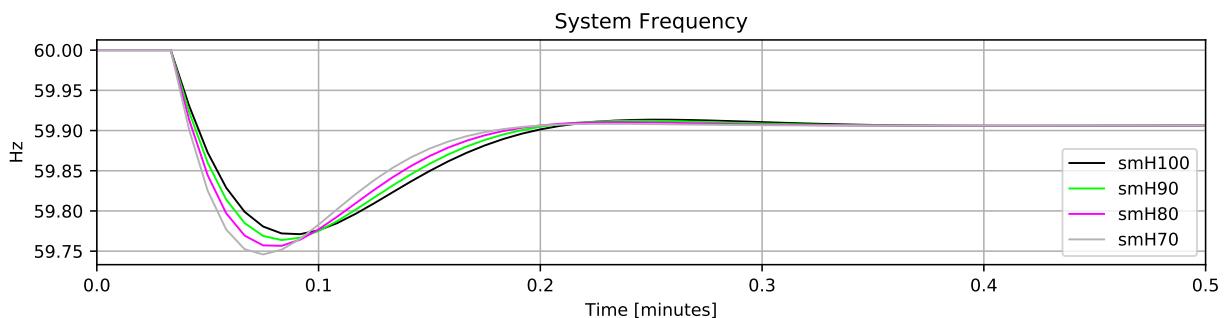


Figure 5.59: Frequency effects of varied system inertia.

Machine responses to the various inertia scalings are shown in Figure 5.60. Results

resemble a reflected frequency response where lower inertia cases result in larger machine response.

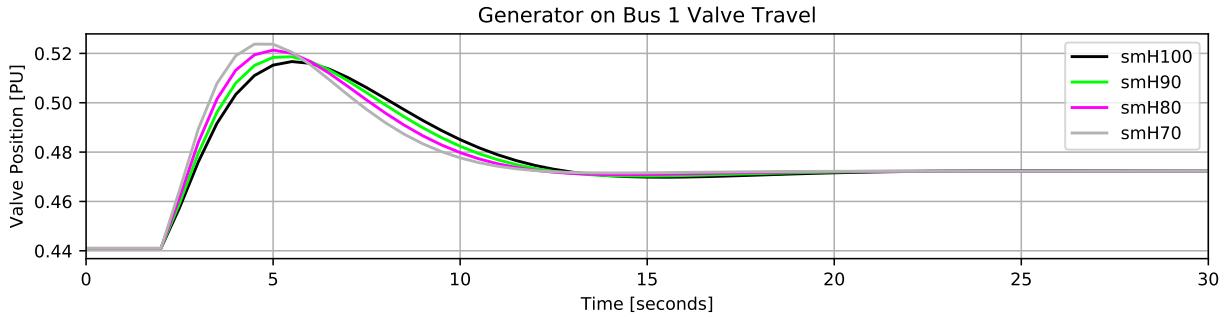


Figure 5.60: Governor response to varied system inertia.

Figure 5.61 shows the resulting changes in system inertia using a perturbation agent to vary inertia during a simulation. Figure 5.62 shows the system frequency response. As expected, lower system inertia leads to faster frequency changes and larger frequency nadirs.

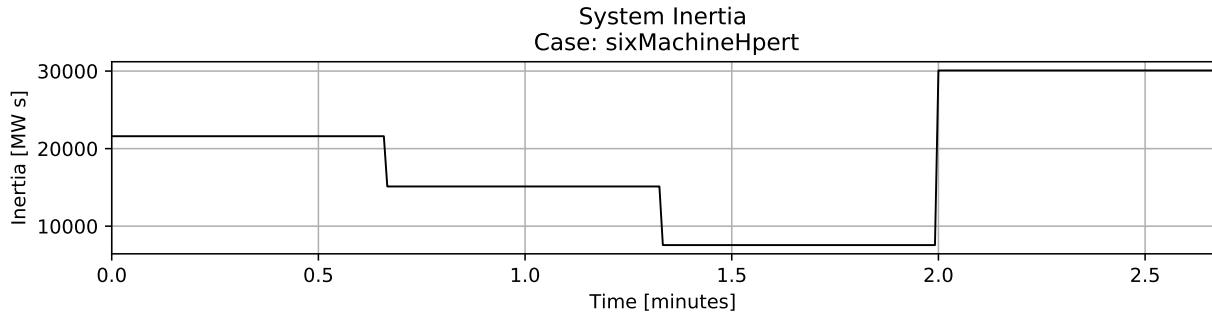


Figure 5.61: Varied system inertia during simulation.

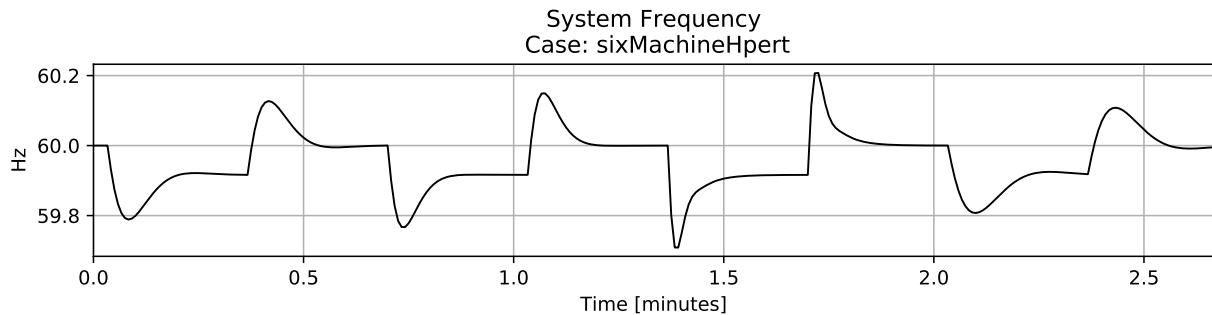


Figure 5.62: System frequency response to varying system inertia.

Damping may be used to alter system frequency dynamic response, but also affects steady state frequency values. System inertia may be changed before or during a simulation. Noticeable effects of varying inertia can be seen during large step type contingencies. PSLTDSim is not particularly well suited for these types of perturbances however, results show that changes in system inertia during simulation are accounted for.

6 Conclusions

6.1 Simulation Conclusions

- Many software engineering goals were successfully met.
- PSLTDSim can use various size and composition system models from PSLF
- Results to ramp and small step type perturbances are comparable, within reason, to transient simulation results
- Many example scenarios have been simulated and described and are available on GitHub.
- Plot functions have been created for system behavior analysis
- PSLTDSim is on PyPI

6.2 Engineering Application Conclusions

- step deadbands increase valve travel.
- Conditional AGC should be used to avoid conflicting area AGC action
- PSLTDSim can be used to tune AGC settings
- Random noise seed value can be adjusted to test varying system scenarios
- Simulations show frequency oscillates between governor deadbands.
- PSLTDSim can be used for switched shunt coordination
- PSLTDSim models can be user modified or created according to need
- System inertia may be altered during the course of simulation

7 Future Work

As with any software project, future work revolves around expansion and refinement. The number of dynamic agents, or governor models, could be expanded without bound. Alternatively, a more refined way of casting un-modeled governors could be devised. Something involving automated one machine infinite bus scenarios, step analysis, and 2nd order approximation has been suggested.

Exponential load models and under-load transformer tap changers are some possible agents that should be accounted for. Powerplant agents that act as plant controllers have been conceptually modeled, but not implemented to their full extent. To better capture voltage changes, and thus reactive power, some form of exciter modeling may be desired, although with the simplification of machine models, this task may prove difficult. Voltage scheduling of generator buses via perturbation agents should be possible, but is untested as of this writing.

PSLTDSim is open-source code that relies on proprietary software for essential functions. To move away from this reliance, a method for creating system models and solving power flows would have to be created. Any changes would have to be incorporated into the way PSLTDSim creates a system mirror, solves a power flow, and updates the mirror. A semi-clear point to perform this break would be when AMQP messages are sent. If PSLTDSim did not rely on PSLF, there would also be no need for AMQP messages to be sent as all code would be PY3. This would not only speed up simulation, but create a more clear flow of events and result in a fully open-source simulation tool.

To accommodate for transient, or oscillatory events, PSLTDSim could be coupled with the ideas presented in [60]. This would essentially involve a variable time step and some set threshold that would automatically switch between TSPF and CTS simulation.

8 Bibliography

- [1] .NET Foundation. (2018). Ironpython overview, [Online]. Available: <https://ironpython.net/>.
- [2] P. M. Anderson and A. A. Fouad, *Power System Control and Stability*, Second Edition. Wiley-Interscience, 2003.
- [3] J. Audenaert, K. Verbeeck, and G. V. Berghe. (2009). Mult-agent based simulation for boarding, CODeS Research Group, [Online]. Available: <https://www.semanticscholar.org/paper/Multi-Agent-Based-Simulation-for-Boarding-Audenaert-Verbeeck-24ba2c3190de5b7162c37e81581b062cda3e4d54>.
- [4] A. Aziz, A. Mto, and A. Stojsevski, “Automatic generation control of multigeneration power system,” Journal of Power and Energy Engineering, 2014.
- [5] J. Carpentier, “‘To be or not to be modern’ that is the question for automatic generation control (point of view of a utility engineer),” International Journal of Electrical Power & Energy Systems, 1985.
- [6] R. W. Cummings, W. Herbsleb, and S. Niemeyer. (2010). Generator governor and information settings webinar, North American Electric Reliability Corporation, [Online]. Available: <https://www.nerc.com/files/gen-governor-info-093010.pdf>.
- [7] F. P. deMello and R. Mills, “Automatic generation control part II - digital control techniques,” IEEE PES Summer Meeting, 1972.
- [8] DEQ. (2004). Montana electric transmission grid: Operation, congestion, and issues, DEQ, [Online]. Available: https://leg.mt.gov/content/publications/Environmental/2004deq_energy_report/transmission.pdf.
- [9] EIA. (2016). U.s. electric system is made up of interconnections and balancing authorities, [Online]. Available: <https://www.eia.gov/todayinenergy/detail.php?id=27152>.
- [10] EIA. (2019). July2019map.png, U.S. Energy Information Administration, [Online]. Available: <https://www.eia.gov/electricity/data/eia860m/>.
- [11] EIA. (2019). U.s. electric system operating data, U.S. Energy Information Administration, [Online]. Available: https://www.eia.gov/realtime_grid/.
- [12] EIA. (2019). U.s. energy mapping system, U.S. Energy Information Administration, [Online]. Available: <https://www.eia.gov/state/maps.php?v=Electricity>.

- [13] E. Ela and J. Kemper, "Wind plant ramping behavior," National Renewable Energy Laboratory, 2009.
- [14] M. D. of Environmental Quality. (2020). Colstrip statistics, DEQ, [Online]. Available: <http://deq.mt.gov/DEQAdmin/mfs/AllColstrip/DEQAdmin/mfs>.
- [15] ERCOT. (2017). Ercot-internconnection_branded.jpg, ERCOT, [Online]. Available: <http://www.ercot.com/news/mediakit/maps>.
- [16] J. H. Eto, J. Undrill, C. Roberts, P. Mackin, and J. Ellis, "Frequency control requirements for reliable interconnection frequency response," Energy Analysis and environmental Impacts Division Lawrence Berkeley National Laboratory, 2018.
- [17] D. Fabozzi and T. Van Cutsem, "Simplified time-domain simulation of detailed long-term dynamic models," IEEE Xplore, 2009.
- [18] FERC, "Essential reliability services and the evolving bulk-power system-primary frequency response," Federal Energy Regulatory Commission, Docket No. RM16-6-000 Order No. 842, Feb. 2018.
- [19] FERC. (2019). About ferc, [Online]. Available: <https://www.ferc.gov/about/about.asp>.
- [20] T. Garnock-Jones and G. M. Roy. (2017). Introduction to pika, [Online]. Available: <https://pika.readthedocs.io/en/stable/>.
- [21] GE Energy, *Mechanics of running psf dynamics*, 2015.
- [22] GeeksforGeeks. (2017). Thread in operating system, GeeksforGeeks, [Online]. Available: <https://www.geeksforgeeks.org/thread-in-operating-system/>.
- [23] General Electric International, Inc, *PSLF User's Manual*, 2016.
- [24] W. B. Gish, "Automatic generation control algorithm - general concepts and application to the watertown energy control center," Bureau of Reclamation Engineering and Research Center, 1980.
- [25] J. D. Glover, M. S. Sarma, and T. J. Overbye, *Power System Analysis & Design*, 5e. Cengage Learning, 2012.
- [26] R. Gonzales. (2019). Pg&e transmission lines caused california's deadliest wildfire, state officials say, NPR, [Online]. Available: <https://www.npr.org/2019/05/15/723753237/pg-e-transmission-lines-caused-californias-deadliest-wildfire-state-officials-sa>.
- [27] M. Goossens, F. Mittelbach, and A. Samarin, *The LATEX Companion*. Addison-Wesley, 1993.

- [28] R. Hallett, "Improving a transient stability control scheme with wide-area synchrophasors and the microwecc, a reduced-order model of the western interconnect," Master's thesis, Montana Tech, 2018.
- [29] E. Heredia, D. Kosterev, and M. Donnelly, "Wind hub reactive resource coordination and voltage control study by sequence power flow," IEEE, 2013.
- [30] J. JMesserly. (2008). Electricity_grid_simple-_north_america.svg, United States Department of Energy, [Online]. Available: https://commons.wikimedia.org/wiki/File:Electricity_grid_simple-_North_America.svg.
- [31] T. Kennedy, S. M. Hoyt, and C. F. Abell, "Variable, non-linear tie-line frequency bias for interconnected systems control," IEEE Transactions on Power Systems, 1988.
- [32] Y. G. Kim, H. Song, and B. Lee, "Governor-response power flow (grpf) based long-term voltage stability simulation," IEEE T&D Asia, 2009.
- [33] G. Kou, P. Markham, S. Hadley, T. King, and Y. Liu, "Impact of governor deadband on frequency response of u.s. eastern interconnection," IEEE Transactions on Smart Grid, 2016.
- [34] D. Kuhlman, *A Python Book: Beginning Python, Advanced Python, and Python Exercises*. 2009.
- [35] P. Kundur, *Power System Stability and Control*. McGraw-Hill, 1994.
- [36] K. H. LaCommare and J. H. Eto, "Understanding the cost of power interruptions to u.s. electricity consumers," Ernest Orlando Lawrence Berkeley National Laboratory, 2004.
- [37] M. Liedtke. (2020). Court approves pg&e's \$23b bankruptcy financing package, AP News, [Online]. Available: <https://apnews.com/b70582ee8d4bb7f781553215612da993>.
- [38] P. Maloney. (2018). What is the value of electric reliability for your operation? [Online]. Available: <https://microgridknowledge.com/power-outage-costs-electric-reliability/>.
- [39] Y. Mobarak, "Effects of the droop speed governor and automatic generation control agc on generator load sharing of power system," International Journal of Applied Power Engineering, 2015.
- [40] NERC, "Frequency response initiative report," North American Electric Reliability Corporation, 2012.
- [41] NERC, "Procedure for ero support of frequency response and frequency bias setting standard," North American Electric Reliability Corporation, 2012.

- [42] NERC, “Standard bal-003-1.1 — frequency response and frequency bias setting,” North American Electric Reliability Corporation, 2015.
- [43] NERC, “Standard bal-001-2 – real power balancing control performance,” North American Electric Reliability Corporation, 2016.
- [44] NERC. (2017). About nerc, [Online]. Available: <https://www.nerc.com/AboutNERC/Pages/default.aspx>.
- [45] NERC. (2017). Nerc interconnections map, [Online]. Available: <https://www.nerc.com/AboutNERC/keyplayers/PublishingImages/NERC%20Interconnections.pdf>.
- [46] NERC, “Bal-002-3 – disturbance control standard – contingency reserve for recovery from a balancing contingency event,” North American Electric Reliability Corporation, 2018.
- [47] NERC, “Frequency response annual analysis,” North American Electric Reliability Corporation, 2018.
- [48] NERC. (2020). Glossary of terms used in nerc reliability standards, NERC, [Online]. Available: https://www.nerc.com/files/glossary_of_terms.pdf.
- [49] NERC Resources Subcommittee, “Balancing and frequency control,” North American Electric Reliability Corporation, 2011.
- [50] NERC Resources Subcommittee, “Bal-001-tre-1 — primary frequency response in the ercot region,” North American Electric Reliability Corporation, 2016.
- [51] P. W. Parfomak, “Physical security of the u.s. power grid: High-voltage transformer substations,” Congressional Research Service, 2014.
- [52] Python Software Foundataion. (2019). About python, [Online]. Available: <https://www.python.org/about/>.
- [53] B. Rand. (2018). Agent-based modeling: What is agent-based modeling? Youtube, [Online]. Available: <https://www.youtube.com/watch?v=FMQbfsoOkGc>.
- [54] C. W. Ross, “Error adaptive control computer for interconnected power systems,” IEEE Transactions on Power Apparatus and Systems, 1966.
- [55] G. van Rossum. (2009). A brief timeline of python, [Online]. Available: <https://python-history.blogspot.com/2009/01/brief-timeline-of-python.html>.
- [56] J. Sanchez-Gasca, M. Donnelly, R. Concepcion, A. Ellis, and R. Elliott, “Dynamic simulation over long time periods with 100% solar generation,” Sandia National Laboratories, SAND2015-11084R, 2015.

- [57] P. W. Sauer, M. A. Pai, and J. H. Chow, *Power System Dynamics and Stability With Synchrophasor Measurement and Power System Toolbox*, Second Edition. John Wiley & Sons Ltd, 2018.
- [58] SciPy developers. (2019). About scipy, [Online]. Available: <https://www.scipy.org/about.html>.
- [59] K. Siegel. (2012). The true cost of power outages, Yale Environment Review, [Online]. Available: <https://environment-review.yale.edu/true-cost-power-outages-0>.
- [60] M. Stajcar, “Power system simulation using an adaptive modeling framework,” Master’s thesis, Montana Tech, 2016.
- [61] C. W. Taylor and R. L. Cresap, “Real-time power system simulation for automatic generation control,” IEEE Transactions on Power Apparatus and Systems, 1976.
- [62] D. Trudnowski, “Properties of the dominant inter-area modes in the wecc interconnect,” Montana Tech, 2012.
- [63] T. Van Cutsem and C. Vournas, *Voltage Stability of Electric Power Systems*, 1st ed. Springer US, 1998.
- [64] WECC. (2015). About wecc, [Online]. Available: <https://www.wecc.org/Pages/AboutWECC.aspx>.

A Six Machine System Details

Relevant values from PSLF tables describing the six machine system used are presented in this appendix. The ‘busd’ table from PSLF describing system buses is shown in Table A.1. The ‘secdd’ table from PSLF describing system lines is shown in Table A.2. The ‘tran’ table from PSLF describing system transformers is shown in Table A.3. The ‘gens’ table from PSLF describing system generators is shown in Table A.4. The ‘load’ table from PSLF describing system loads is shown in Table A.5. The ‘shunt’ table from PSLF describing system shunts is shown in Table A.6. The dyd file used for validation is shown in Figure A.1.

Table A.1: Six machine bus table.

BUS-NO	NAME	KV	TP	VSCHEd	V-PU	DEG	AREA
6	6	138.00	1	1.00	0.9537	-15.91	1
7	7	138.00	1	1.00	0.9533	-16.07	1
8	8	138.00	1	1.00	0.9532	-16.52	1
9	9	138.00	1	1.00	0.9533	-16.54	2
10	10	138.00	1	1.00	0.9543	-16.14	2
11	11	138.00	1	1.00	0.9549	-15.78	2
1	1	22.00	0	1.00	1.0000	0.00	1
2	2	22.00	2	1.00	1.0000	11.41	1
3	3	22.00	2	1.00	1.0000	1.27	2
4	4	22.00	2	1.00	1.0000	1.27	2
5	5	22.00	2	1.00	1.0000	-10.72	2

Table A.2: Six machine line table.

FROM	FNAME	FKV	TO	TNAME	TKV	CK	SE	R-PU	X-PU	B-PU	AF	AT
6	6	138.00	7	7	138.00	1	1	0.0001	0.001	0.0018	1	1
7	7	138.00	8	8	138.00	1	1	0.0001	0.001	0.0018	1	1
8	8	138.00	9	9	138.00	1	1	0.0001	0.001	0.0018	1	2
8	8	138.00	9	9	138.00	2	1	0.0001	0.001	0.0018	1	2
8	8	138.00	9	9	138.00	3	1	0.0001	0.001	0.0018	1	2
11	11	138.00	10	10	138.00	1	1	0.0001	0.001	0.0018	2	2
10	10	138.00	9	9	138.00	1	1	0.0001	0.001	0.0018	2	2

Table A.3: Six machine transformer table.

FROM	FNAME	FKV	TO	TNAME	TKV	MVA	VNOMF	VNOMT	R	X	BMAG	AREA
1	1	22.00	6	6	138.00	100.00	22.00	138.00	0.00	0.10	0.00	1
2	2	22.00	7	7	138.00	100.00	22.00	138.00	0.00	0.10	0.00	1
3	3	22.00	11	11	138.00	100.00	22.00	138.00	0.00	0.10	0.00	2
4	4	22.00	11	11	138.00	100.00	22.00	138.00	0.00	0.10	0.00	2
5	5	22.00	10	10	138.00	100.00	22.00	138.00	0.00	0.10	0.00	2

Table A.4: Six machine generator table.

BUS-NO	NAME1	KV1	ID	PGEN	QGEN	IREG	AREA	MBASE	PMAX
1	1	22.00	1	261.40	82.90	1	1	900.00	1000.00
2	2	22.00	1	220.00	77.10	2	1	900.00	1000.00
2	2	22.00	2	220.00	77.10	2	1	900.00	1000.00
3	3	22.00	1	280.00	87.10	3	2	900.00	1000.00
4	4	22.00	1	280.00	87.10	4	2	900.00	1000.00
5	5	22.00	1	90.00	49.90	5	2	900.00	1000.00

Table A.5: Six machine load table.

BUS-NO	NAME	KV	ID	ST	PLOAD	QLOAD	AREA
8	8	138.00	1	1	600.00	100.00	1
9	9	138.00	1	1	750.00	100.00	2

Table A.6: Six machine shunt table.

FROM	FNAME	FKV	ID	CK	ST	G-PU	B-PU	AREA
8	8	138.00	1	1	1	0.00	1.00	1
8	8	138.00	2	2	1	0.00	0.50	1
8	8	138.00	3	3	0	0.00	0.50	1
8	8	138.00	4	4	0	0.00	0.50	1
9	9	138.00	1	1	1	0.00	1.00	2
9	9	138.00	2	2	0	0.00	0.50	2
9	9	138.00	3	3	0	0.00	0.50	2
9	9	138.00	4	4	0	0.00	0.50	2

```

1 # Six Machine, all gens and govs the same
2 # exciters default settings
3
4 # Metering of frequency, current, and voltages
5 fmeta 1 "1" 22.00 "1 " : #9 0 1
6 ameta 1 "1" 22.00 "1 " : 0
7 vmeta 1 "1" 22.00 "1 " : 0
8
9 # current meters
10 imetr 6 ! ! ! 7 ! ! ! : #9 0
11 imetr 7 ! ! ! 8 ! ! ! : #9 0
12 imetr 8 ! ! ! 9 ! ! ! : #9 0
13 imetr 11 ! ! ! 10 ! ! ! : #9 0
14 imetr 10 ! ! ! 9 ! ! ! : #9 0
15
16 # loads (Using wlwscc on any load sets the dynamics characteristics of all loads.)
17 wlwscc 9 "9" 138.0 "1 " : #9 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0
18
19 # generators
20 genrou 1 "1" 22.00 "1 " : #9 mva=900.00 "tpdo" 6.50 "tppdo" 0.079 "tpqo" 0.53 "tppqo" 0.072 "h" 4 "d"
  ↳ 0.0000 "ld" 1.24 "lq" 1.22 "lpd" 0.23 "lpq" 0.36000 "lppd" 0.17 "ll" 0.14 "s1" 0.173 "s12" 0.447 "ra"
  ↳ 0.0000 "rcomp" 0.0000 "xcomp" 0.0000 "accel" 1.0
21 genrou 2 "2" 22.00 "1 " : #9 mva=900.00 "tpdo" 6.50 "tppdo" 0.079 "tpqo" 0.53 "tppqo" 0.072 "h" 4 "d"
  ↳ 0.0000 "ld" 1.24 "lq" 1.22 "lpd" 0.23 "lpq" 0.36000 "lppd" 0.17 "ll" 0.14 "s1" 0.173 "s12" 0.447 "ra"
  ↳ 0.0000 "rcomp" 0.0000 "xcomp" 0.0000 "accel" 1.0
22 genrou 2 "2" 22.00 "2 " : #9 mva=900.00 "tpdo" 6.50 "tppdo" 0.079 "tpqo" 0.53 "tppqo" 0.072 "h" 4 "d"
  ↳ 0.0000 "ld" 1.24 "lq" 1.22 "lpd" 0.23 "lpq" 0.36000 "lppd" 0.17 "ll" 0.14 "s1" 0.173 "s12" 0.447 "ra"
  ↳ 0.0000 "rcomp" 0.0000 "xcomp" 0.0000 "accel" 1.0
23 genrou 3 "3" 22.00 "1 " : #9 mva=900.00 "tpdo" 6.50 "tppdo" 0.079 "tpqo" 0.53 "tppqo" 0.072 "h" 4 "d"
  ↳ 0.0000 "ld" 1.24 "lq" 1.22 "lpd" 0.23 "lpq" 0.36000 "lppd" 0.17 "ll" 0.14 "s1" 0.173 "s12" 0.447 "ra"
  ↳ 0.0000 "rcomp" 0.0000 "xcomp" 0.0000 "accel" 1.0
24 genrou 4 "4" 22.00 "1 " : #9 mva=900.00 "tpdo" 6.50 "tppdo" 0.079 "tpqo" 0.53 "tppqo" 0.072 "h" 4 "d"
  ↳ 0.0000 "ld" 1.24 "lq" 1.22 "lpd" 0.23 "lpq" 0.36000 "lppd" 0.17 "ll" 0.14 "s1" 0.173 "s12" 0.447 "ra"
  ↳ 0.0000 "rcomp" 0.0000 "xcomp" 0.0000 "accel" 1.0
25 genrou 5 "5" 22.00 "1 " : #9 mva=900.00 "tpdo" 6.50 "tppdo" 0.079 "tpqo" 0.53 "tppqo" 0.072 "h" 4 "d"
  ↳ 0.0000 "ld" 1.24 "lq" 1.22 "lpd" 0.23 "lpq" 0.36000 "lppd" 0.17 "ll" 0.14 "s1" 0.173 "s12" 0.447 "ra"
  ↳ 0.0000 "rcomp" 0.0000 "xcomp" 0.0000 "accel" 1.0
26
27 # exciters
28 sexs 1 "1" 22.00 "1 " : #1 0.1 10.0 100.0 0.05 -5.0 5.0 0.08 0.0 -5.0 5.0 0.0
29 sexs 2 "2" 22.00 "1 " : #1 0.1 10.0 100.0 0.05 -5.0 5.0 0.08 0.0 -5.0 5.0 0.0
30 sexs 2 "2" 22.00 "2 " : #1 0.1 10.0 100.0 0.05 -5.0 5.0 0.08 0.0 -5.0 5.0 0.0
31 sexs 3 "3" 22.00 "1 " : #1 0.1 10.0 100.0 0.05 -5.0 5.0 0.08 0.0 -5.0 5.0 0.0
32 sexs 4 "4" 22.00 "1 " : #1 0.1 10.0 100.0 0.05 -5.0 5.0 0.08 0.0 -5.0 5.0 0.0
33 sexs 5 "5" 22.00 "1 " : #1 0.1 10.0 100.0 0.05 -5.0 5.0 0.08 0.0 -5.0 5.0 0.0
34
35 # governors
36 tgov1 1 "1" 22.00 "1 " : #1 mwcap=800.0000 0.050000 0.4 1.000000 0.0 3.0000 10.0000 0.0
37 tgov1 2 "2" 22.00 "1 " : #1 mwcap=800.0000 0.050000 0.4 1.000000 0.0 3.0000 10.0000 0.0
38 tgov1 2 "2" 22.00 "2 " : #1 mwcap=800.0000 0.050000 0.4 1.000000 0.0 3.0000 10.0000 0.0
39 tgov1 3 "3" 22.00 "1 " : #1 mwcap=800.0000 0.050000 0.4 1.000000 0.0 3.0000 10.0000 0.0
40 tgov1 4 "4" 22.00 "1 " : #1 mwcap=800.0000 0.050000 0.4 1.000000 0.0 3.0000 10.0000 0.0

```

Figure A.1: Dyd file used in six machine validations.

B Code Examples

This appendix is used to present code examples too large for inclusion in the body of the text. Some examples span multiple pages. To adhere to document format requirements, the description of each code example is presented after the corresponding code figure.

```

1  # Format of required info for batch runs.
2
3  debug = 0
4  AMQPdebug = 0
5  debugTimer = 0
6
7  simNotes = """
8    AGC TUNING (no delay)
9    Delay over response test
10   Loss of generation in area 1 at t=2
11   Delayed action by area 2
12   AGC in both areas
13 """
14
15 # Simulation Parameters Dictionary
16 simParams = {
17     'timeStep': 1.0, # seconds
18     'endTime': 60.0*8, # seconds
19     'slackTol': 1, # MW
20     'PY3msgGroup' : 3, # number of Agent msgs per AMQP msg
21     'IPYmsgGroup' : 60, # number of Agent msgs per AMQP msg
22     'Hinput' : 0.0, # MW*sec of entire system, if !> 0.0, will be calculated in code
23     'Dsys' : 0.0, # Damping
24     'fBase' : 60.0, # System F base in Hertz
25     'freqEffects' : True, # w in swing equation will not be assumed 1 if true
26     # Mathematical Options
27     'integrationMethod' : 'rk45',
28     # Data Export Parameters
29     'fileDirectory' : "\\\delme\\\200109-delayScenario1\\", # relative path from cwd
30     'fileName' : 'SixMachineDelayStep1',
31     'exportFinalMirror': 1, # Export mirror with all data
32     'exportMat': 1, # if IPY: requies exportDict == 1 to work
33     'exportDict' : 0, # when using python 3 no need to export dicts.
34     'deleteInit' : 0, # Delete initialized mirror
35     'assumedV' : 'Vsched', # assumed voltage - either Vsched or Vinit
36     'logBranch' : True,
}

```

```

37
38 savPath = r"C:\LTD\pslf_systems\sixMachine\sixMachineTrips.sav"
39 dydPath = [r"C:\LTD\pslf_systems\sixMachine\sixMachineDelay.dyd"]
40 ltdPath = r".\testCases\200109-delayScenario1\sixMachineDelayStep1.ltd.py"

```

Figure B.1: An example of a full .py simulation file.

```

1 # Format of required info for batch runs.
2 debug = 0
3 AMQPdebug = 0
4 debugTimer = 0
5
6 simNotes = """
7 agc with deadband and nz, area 2 perturbation TLB 4
8 """
9
10 # Simulation Parameters Dictionary
11 simParams = {
12     'timeStep': 1.0,
13     'endTime': 60.0*10,
14     'slackTol': 1,
15     'PY3msgGroup' : 3,
16     'IPYmsgGroup' : 60,
17     'Hinput' : 0.0, # MW*sec of entire system, if !> 0.0, will be calculated in code
18     'Dsys' : 0.0, # Untested
19     'fBase' : 60.0, # System F base in Hertz
20     'freqEffects' : True, # w in swing equation will not be assumed 1 if true
21     # Mathematical Options
22     'integrationMethod' : 'rk45',
23     # Data Export Parameters
24     'fileDirectory' : "\\\delme\\200325-smFinal\\", # relative path from cwd
25     'fileName' : 'smAGCt4Ex1',
26     'exportFinalMirror': 1, # Export mirror with all data
27     'exportMat': 1, # if IPY: requires exportDict == 1 to work
28     'exportDict' : 0, # when using python 3 no need to export dicts.
29     'deleteInit' : 0, # Delete initialized mirror
30     'assumedV' : 'Vsched', # assumed voltage - either Vsched or Vinit
31     'logBranch' : True,
32 }
33
34 savPath = r"C:\LTD\pslf_systems\sixMachine\sixMachineLTD.sav"

```

```

35 dydPath = [r"C:\LTD\pslf_systems\sixMachine\sixMachineLTD.dyd"]
36 ltdPath = r".\testCases\200325-smFinals\smAGCt4Ex1.ltd.py"

```

Figure B.2: Required .py file for external AGC event with conditional ACE.

```

1 # Perturbances
2 mirror.sysPerturbances = [
3     'gen 5 : step Pm 2 -150 rel',
4 ]
5
6 mirror.NoiseAgent = ltd.perturbation.LoadNoiseAgent(mirror, 0.05, walk=True, delay=0,
7 → damping=0, seed=11)
8
9 # Balancing Authorities
10 mirror.sysBA = {
11     'BA1':{
12         'Area':1,
13         'B': "0.9 : permax", # MW/0.1 Hz
14         'AGCActionTime': 30.00, # seconds
15         'ACEgain' : 1.0,
16         'AGCType':'TLB : 4', # Tie-Line Bias
17         'UseAreaDroop' : False,
18         'AreaDroop' : 0.05,
19         'IncludeIACE' : True,
20         'IACEconditional': True,
21         'IACEwindow' : 30, # seconds - size of window - 0 for non window
22         'IACEScale' : 1/5,
23         'IACEdeadband' : 0, # Hz
24         'ACEFiltering': 'PI : 0.04 0.0001',
25         'AGCDeadband' : None, # MW? -> not implemented
26         'GovDeadbandType' : 'nldroop', # step, None, ramp, nldroop
27         'GovDeadband' : .036, # Hz
28         'GovAlpha' : 0.016, # Hz - for nldroop
29         'GovBeta' : 0.036, # Hz - for nldroop
30         'CtrlGens': ['gen 1 : 0.5 : rampA',
31                      'gen 2 1 : 0.5 : rampA',
32                      ],
33     },
34     'BA2':{
35         'Area':2,
36         'B': "0.9 : permax", # MW/0.1 Hz
37         'AGCActionTime': 45.00, # seconds

```

```

37     'ACEgain' : 1.0,
38     'AGCType':'TLB : 4', # Tie-Line Bias
39     'UseAreaDroop' : False,
40     'AreaDroop' : 0.05,
41     'IncludeIACE' : True,
42     'IACEconditional': True,
43     'IACEwindow' : 45, # seconds - size of window - 0 for non window
44     'IACEscale' : 1/3,
45     'IACEdeadband' : 0, # Hz
46     'ACEFiltering': 'PI : 0.04 0.0001',
47     'AGCDeadband' : None, # MW? -> not implemented
48     'GovDeadbandType' : 'nldroop', # step, None, ramp, nldroop
49     'GovDeadband' : .036, # Hz
50     'GovAlpha' : 0.016, # Hz - for nldroop
51     'GovBeta' : 0.036, # Hz - for nldroop
52     'CtrlGens': ['gen 3 : 1.0 : rampA'],
53   },
54 }
```

Figure B.3: Required .ltd file for external AGC event with conditional ACE.

```

1 mirror.NoiseAgent = ltd.perturbation.LoadNoiseAgent(mirror, 0.05, walk=True, delay=0,
2   ↳ damping=0, seed=11)
3
4 # Balancing Authorities
5 mirror.sysBA = {
6   'BA1':{
7     'Area':1,
8     'B': "0.9 : permax", # MW/0.1 Hz
9     'AGCActionTime': 30.00, # seconds
10    'ACEgain' : 1.0,
11    'AGCType':'TLB : 4', # Tie-Line Bias
12    'UseAreaDroop' : False,
13    'AreaDroop' : 0.05,
14    'IncludeIACE' : True,
15    'IACEconditional': True,
16    'IACEwindow' : 30, # seconds - size of window - 0 for non window
17    'IACEscale' : 1/5,
18    'IACEdeadband' : 0, # Hz
19    'ACEFiltering': 'PI : 0.04 0.0001',
20    'AGCDeadband' : None, # MW? -> not implemented
21    'GovDeadbandType' : 'nldroop', # step, None, ramp, nldroop
22  }
23 }
```

```

21     'GovDeadband' : .036, # Hz
22     'GovAlpha' : 0.016, # Hz - for nldroop
23     'GovBeta' : 0.036, # Hz - for nldroop
24     'CtrlGens': ['gen 1 : 0.5 : rampA',
25                   'gen 2 1 : 0.5 : rampA',
26                   ]
27   },
28   'BA2':{
29     'Area':2,
30     'B': "0.9 : permax", # MW/0.1 Hz
31     'AGCActionTime': 45.00, # seconds
32     'ACEgain' : 1.0,
33     'AGCType':'TLB : 4', # Tie-Line Bias
34     'UseAreaDroop' : False,
35     'AreaDroop' : 0.05,
36     'IncludeIACE' : True,
37     'IACEconditional': True,
38     'IACEwindow' : 45, # seconds - size of window - 0 for non window
39     'IACEScale' : 1/3,
40     'IACEdeadband' : 0, # Hz
41     'ACEFiltering': 'PI : 0.04 0.0001',
42     'AGCDeadband' : None, # MW? -> not implemented
43     'GovDeadbandType' : 'nldroop', # step, None, ramp, nldroop
44     'GovDeadband' : .036, # Hz
45     'GovAlpha' : 0.016, # Hz - for nldroop
46     'GovBeta' : 0.036, # Hz - for nldroop
47     'CtrlGens': ['gen 3 : 1.0 : rampA',]
48   },
49 }
50
51 # Load and Generation Cycle Agents
52 mirror.sysGenerationControl = {
53   'BPATDispatch' : {
54     'Area': 1,
55     'startTime' : 2,
56     'timeScale' : CTRLtimeScale,
57     'rampType' : 'per', # relative percent change
58     'CtrlGens': [
59       "gen 1 : 0.5",
60       "gen 2 1 : 0.5",
61     ],
62     # Data from: 12/11/2019 PACE
63     'forecast' : [
64       #(time , Precent change from previous value)

```

```

65     (0, 0.0),
66     (1, 5.8),
67     (2, 8.8),
68     (3, 9.9),
69     (4, 4.0),
70   ],
71 }, #end of generation controller def
72 'CAISODispatch' : {
73   'Area': 2,
74   'startTime' : 2,
75   'timeScale' : CTRLtimeScale,
76   'rampType' : 'per', # relative percent change
77   'CtrlGens': [
78     "gen 4 : 1.0",
79   ],
80   # Data from: 12/11/2019 PACE
81   'forecast' : [
82     #(time , Precent change from previous value)
83     (0, 0.0),
84     (1, 0.7),
85     (2, 7.5),
86     (3, 11.2),
87     (4, 4.4),
88   ],
89 }, #end of generation controller def
90 }
91
92 mirror.sysLoadControl = {
93   'BPATDemand' : {
94     'Area': 1,
95     'startTime' : 2,
96     'timeScale' : CTRLtimeScale,
97     'rampType' : 'per', # relative percent change
98     # Data from: 12/11/2019 BPAT
99     'demand' : [
100       #(time , Precent change from previous value)
101       (0, 0.000),
102       (1, 3.2),
103       (2, 8.2),
104       (3, 9.3),
105       (4, 3.8),
106     ] ,
107   }, # end of demand agent def
108   'CAISODemand' : {

```

```

109     'Area': 2,
110     'startTime' : 2,
111     'timeScale' : CTRLtimeScale,
112     'rampType' : 'per', # relative percent change
113     # Data from: 12/11/2019 CAISO
114     'demand' : [
115         #(time , Precent change from previous value)
116         (0, 0.000),
117         (1, 3.0),
118         (2, 7.0),
119         (3, 10.5),
120         (4, 4.4),
121     ] ,
122 },# end of demand load control definition
123 }# end of loac control definitions
124
125 # Definite Time Controller Definitions
126 mirror.DTCdict = {
127     'bus8caps' : {
128         'RefAgents' : {
129             'ra1' : 'bus 8 : Vm',
130             'ra2' : 'branch 8 9 1 : Qbr', # branches defined from, to, ckID
131         },# end Reference Agents
132         'TarAgents' : {
133             'tar1' : 'shunt 8 2 : St',
134             'tar2' : 'shunt 8 3 : St',
135             'tar3' : 'shunt 8 4 : St',
136             'tar4' : 'shunt 8 5 : St',
137             'tar5' : 'shunt 8 6 : St',
138         }, # end Target Agents
139         'Timers' : {
140             'set' :{ # set shunts
141                 'logic' : "(ra1 < 1.0)", # or (ra2 < -26)",
142                 'actTime' : 30, # seconds of true logic before act
143                 'act' : "anyOFFTar = 1", # set any target off target = 1
144             },# end set
145             'reset' :{ # reset shunts
146                 'logic' : "(ra1 > 1.04)",# or (ra2 > 26)",
147                 'actTime' : 30, # seconds of true logic before act
148                 'act' : "anyONTar = 0", # set any target On target = 0
149             },# end reset
150             'hold' : 90, # minimum time between actions
151         }, # end timers
152     },# end bus8caps

```

```

153 'bus9caps' : {
154     'RefAgents' : {
155         'ra1' : 'bus 9 : Vm',
156         'ra2' : 'branch 8 9 1 : Qbr', # branches defined from, to, ckID
157     }, # end Reference Agents
158     'TarAgents' : {
159         'tar1' : 'shunt 9 2 : St',
160         'tar2' : 'shunt 9 3 : St',
161         'tar3' : 'shunt 9 4 : St',
162         'tar4' : 'shunt 9 5 : St',
163         'tar5' : 'shunt 9 6 : St',
164     }, # end Target Agents
165     'Timers' : {
166         'set' : { # set shunts
167             'logic' : "(ra1 < 1.0)",
168             'actTime' : 45, # seconds of true logic before act
169             'act' : "anyOFFTar = 1", # set any target off target = 1
170         }, # end set
171         'reset' : { # reset shunts
172             'logic' : "(ra1 > 1.04)",
173             'actTime' : 45, # seconds of true logic before act
174             'act' : "anyONTar = 0", # set any target On target = 0
175         }, # end reset
176         'hold' : 120, # minimum time between actions
177     }, # end timers
178 }, # end bus8caps
179 }# end DTCdict

```

Figure B.4: Required .ltd file for forecast demand scenario with noise and deadbands.

C Large Tables

This appendix is used to present large tables too distracting for inclusion in their respective sections.

Table C.1: Balancing authority dictionary input information.

Key	Type	Units	Example	Description
B	String	MW/0.1Hz	"1.0 : permax"	Describes the frequency bias scaling factor B used in the ACE calculation. Various Options exist.
AGCActionTime	Float	Seconds	5	Time between AGC dispatch messages.
AGCType	String	-	"TLB : 2"	Dictates which AGC routine to use and type specific options.
UseAreaDroop	Boolean	-	FALSE	If True, all governed generators under BA control will use the area droop.
AreaDroop	Float	Hz/MW	0.05	Droop value to use if 'UseAreaDroop' is True.
IncludeIACE	Boolean	-	TRUE	If True, include IACE in ACE calculation
IACEconditional	Boolean	-	FALSE	Adds IACE to ACE if signs of deltaraw and IACE match.
IACEwindow	Integer	Seconds	60	Defines the length of moving integration window to use in IACE. If set to 0, integration takes place for all time.
IACEScale	Float	-	0.0167	Value used to scale IACE.
IACEDeadband	Float	Hz	0.036	Absolute value of system frequency where IACE will not be calculated below.
ACEFiltering	String	-	PI : 0.03 0.001'	String used to dictate which filter agent is created and filter specific parameters.
AGCDeadband	Float	MW	1.5	Value of ACE to ignore sending in AGC dispatch. Not implemented as of this writing.
GovDeadbandType	String	-	'step'	Type of deadband to be applied to area governors.
GovDeadband	Float	Hz	0.036	Absolute value of system frequency that governors will not respond below.
GovAlpha	Float	Hz	0.016	Specific to 'NLdroop' type of deadband. Specifies lower bound of non-linear droop.
GovBeta	Float	Hz	0.036	Specific to 'NLdroop' type of deadband. Specifies upper bound of non-linear droop.
CtrlGens	List of Strings	-	-	List of generators, participation factor, and dispatch signal type.

Table C.2: Simulation parameters dictionary input information.

Key	Type	Units	Example	Description
timeStep	float	Seconds	1	Simulated time between power-flow solutions
endTime	float	Seconds	1800	Number of seconds simulation is to run for.
slackTol	float	MW	0.5	MW Value that slack error must be below for returned solution to be accepted.
PY3msgGroup	integer	-	3	Number of messages to combine into one AMQP message for PY3 to IPY communication.
IPYmsgGroup	integer	-	60	Number of messages to combine into one AMQP message for IPY to PY3 communication.
Hinput	float	MW sec	0	Value to use for total system inertia. Units are MW*sec. If set to 0.0, system inertia will be calculated from the given .sav information.
Dsys	float	PU	0	Value of system damping used in swing equation and governor models. While this option is available, it is untested and typically set to 0.0.
fBase	float	Hz	60	Value of base system frequency.
freqEffects	boolean	-	True	If True, the ω used in the swing equation will be the current system frequency. If this is set to False then ω will be set equal to 1 for the swing equation calculation
integrationMethod	string	-	'rk45'	This option defines how the swing equation is integrated to find current frequency. Valid options are 'rk45', 'ab', and 'euler'. The default is the 'euler' method which is a simple forward Euler integration. The 'ab' option uses a two step Adams-Bashforth method and the 'rk45' option uses the scipy <code>solve_ivp</code> function that utilizes an explicit Runge-Kutta 4(5) method.
fileDirectory	string	-	"\\delme\\\"	This is a relative path location from the folder where PSLTDSim is executed in which the output files are saved to.
fileName	string	-	"SimTest"	This is the name used to save files.
exportFinalMirror	int	-	1	If this value is 1 a final system mirror will be exported. If this value is 0 no final mirror will be exported.
exportMat	int	-	1	If this value is 1 a MATLAB .mat file will be exported. If this value is 0 no MATLAB .mat file will be exported.

D Detailed Valve Travel Results

This appendix is used to present detailed valve travel plots from various deadband scenarios that were used to populate data presented in Table 5.2.

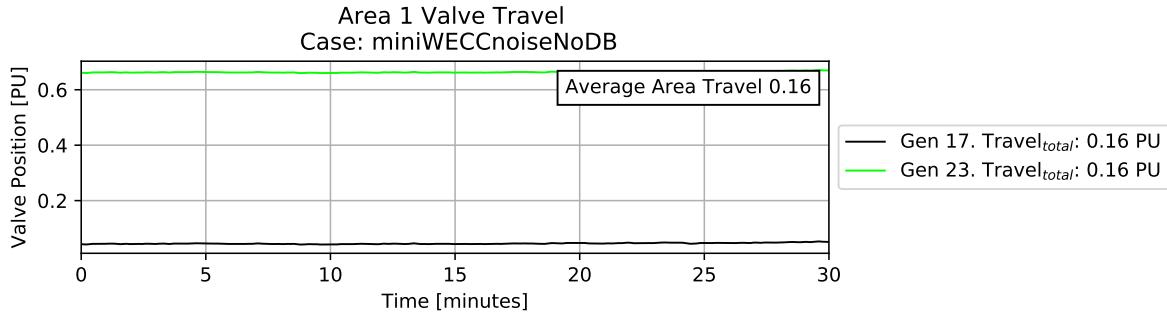


Figure D.1: Area 1 valve travel using no deadband.

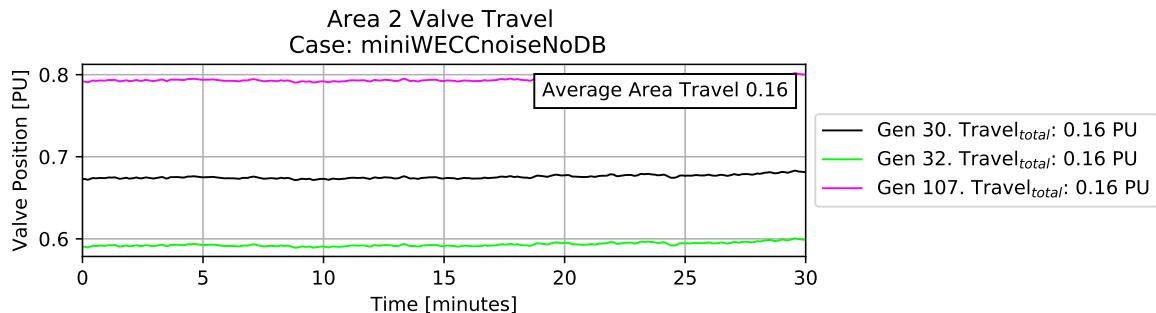


Figure D.2: Area 2 valve travel using no deadband.

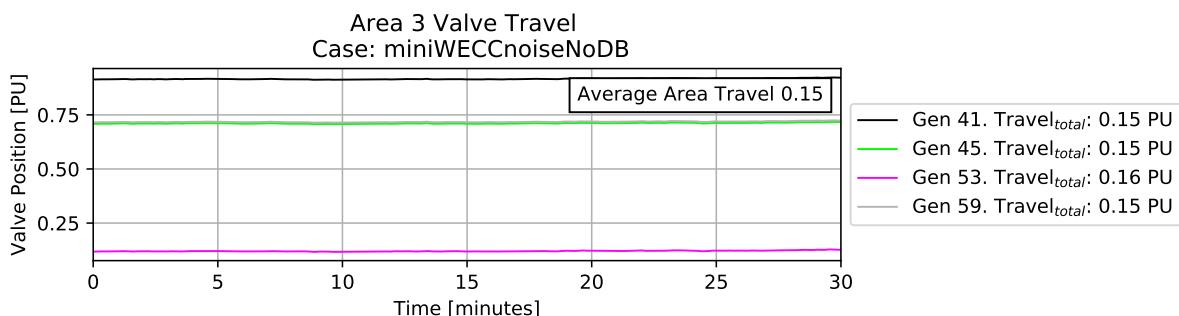


Figure D.3: Area 3 valve travel using no deadband.

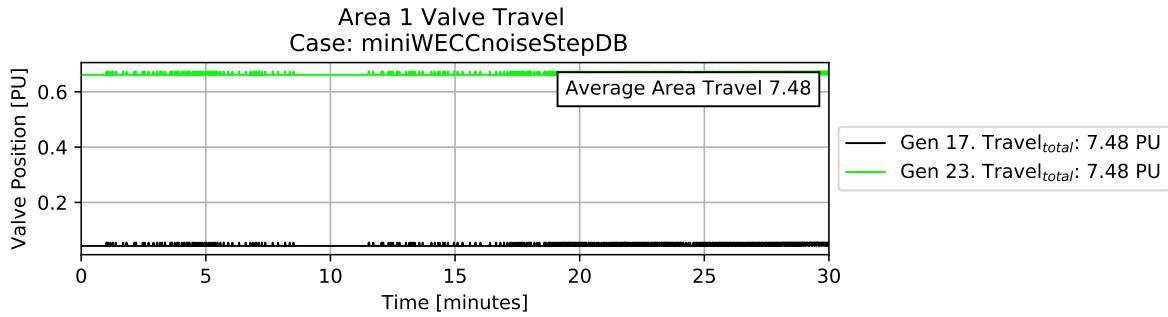


Figure D.4: Area 1 valve travel using a step deadband.

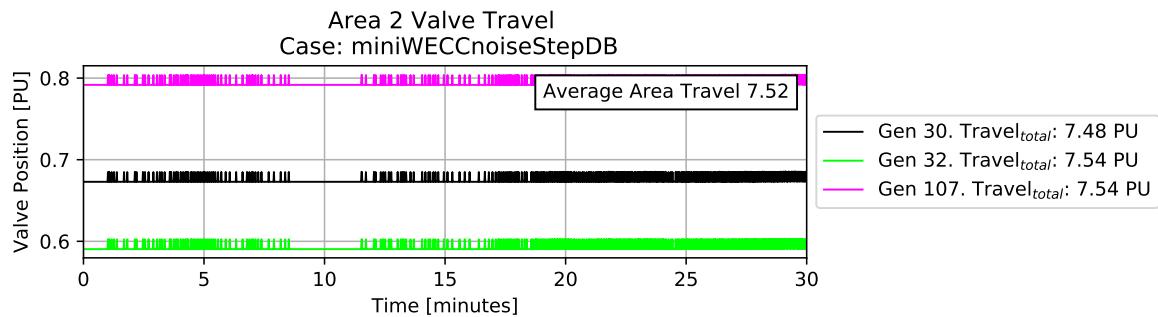


Figure D.5: Area 2 valve travel using a step deadband.

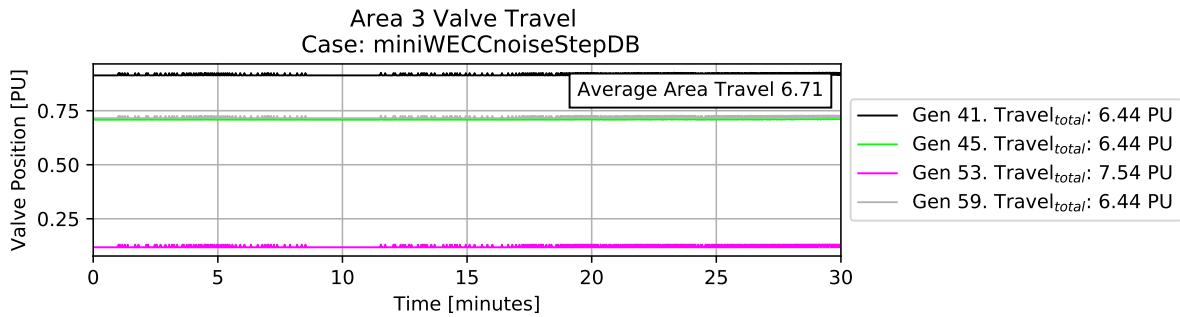


Figure D.6: Area 3 valve travel using a step deadband.

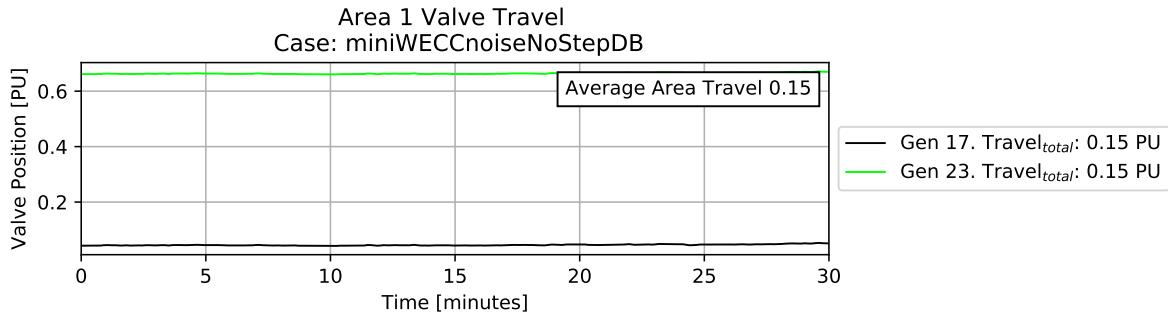


Figure D.7: Area 1 valve travel using a no-step deadband.

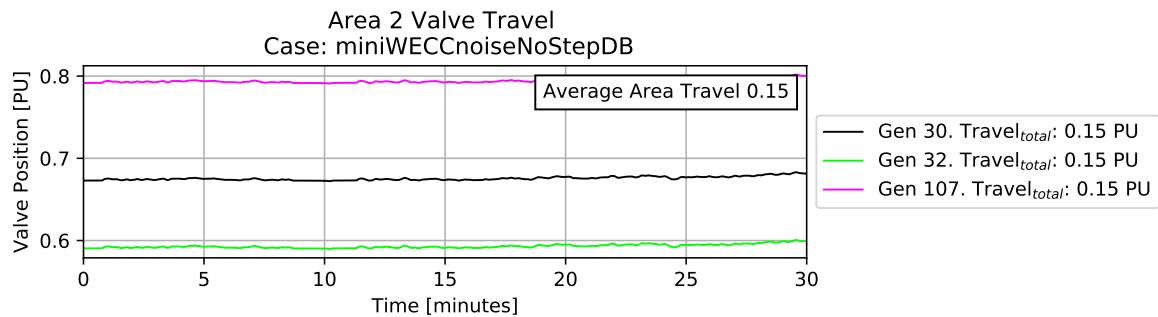


Figure D.8: Area 2 valve travel using a no-step deadband.

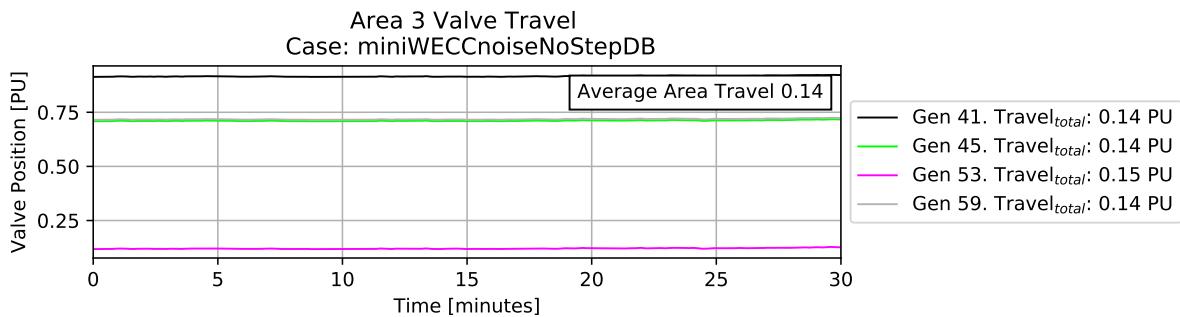


Figure D.9: Area 3 valve travel using a no-step deadband.

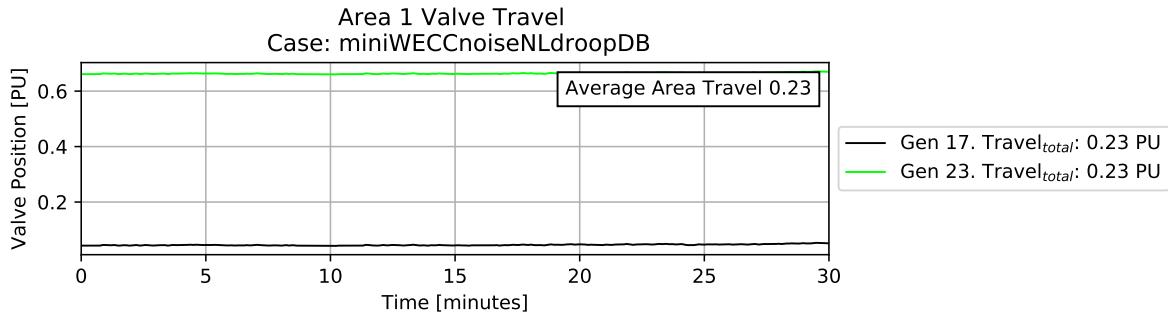


Figure D.10: Area 1 valve travel using a non-linear droop deadband.

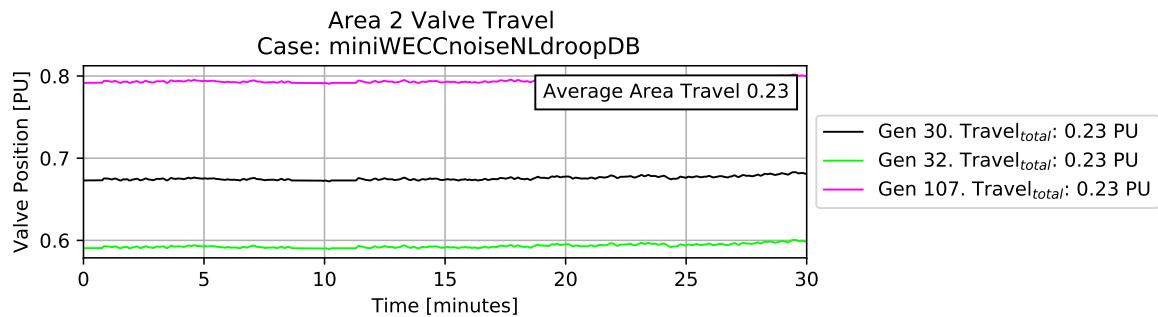


Figure D.11: Area 2 valve travel using a non-linear droop deadband.

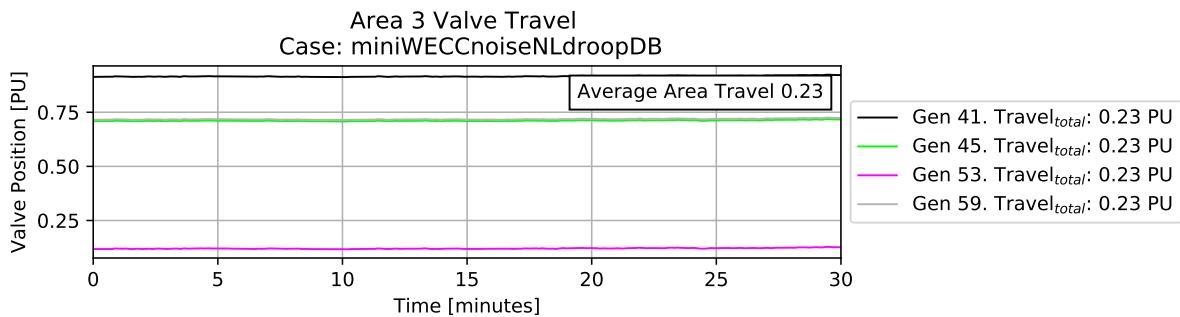


Figure D.12: Area 3 valve travel using a non-linear droop deadband.

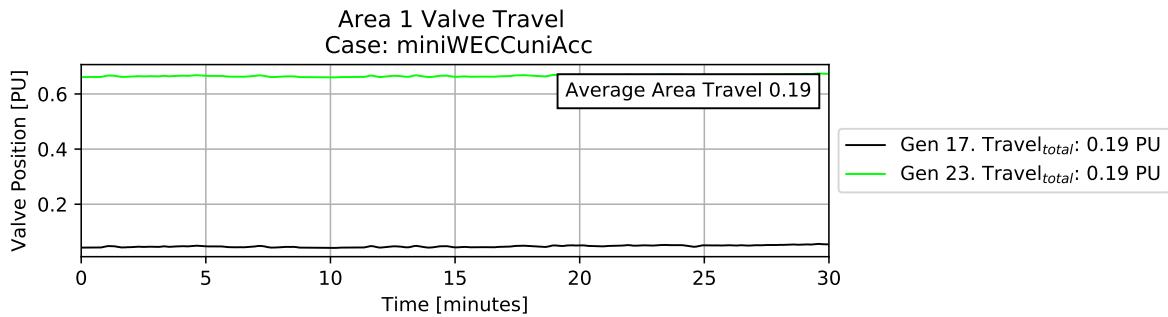


Figure D.13: Area 1 valve travel in a non-homogeneous deadband system.

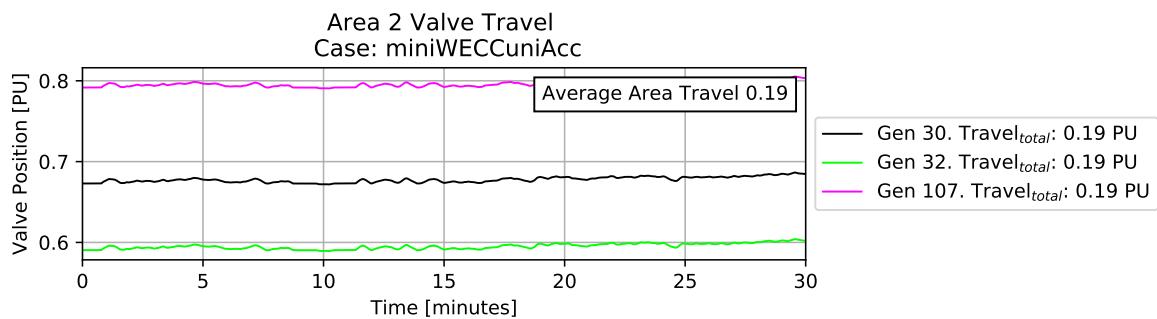


Figure D.14: Area 2 valve travel in a non-homogeneous deadband system.

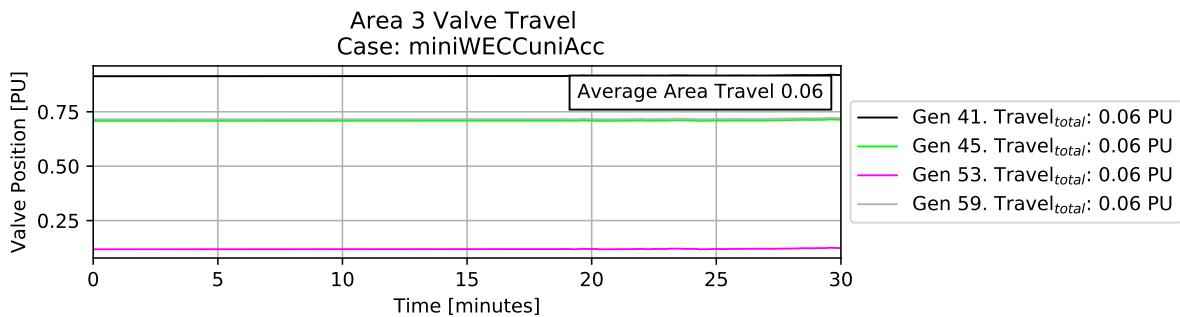


Figure D.15: Area 3 valve travel in a non-homogeneous deadband system.

E Additional AGC Results

This appendix is used to present figures from AGC base case, tuning, and deadband noise results involving Area 2, which are largely similar to Area 1 results. Conditional AGC response to an area 2 event is also presented.

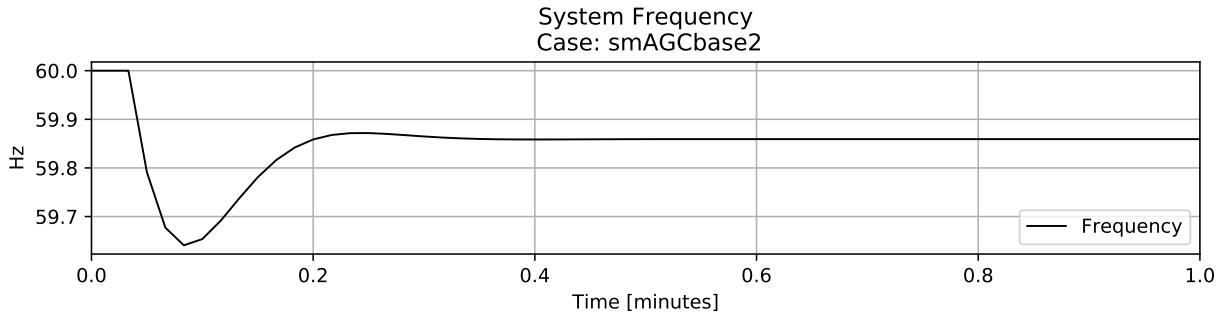


Figure E.1: Frequency response to generation loss event in area 2.

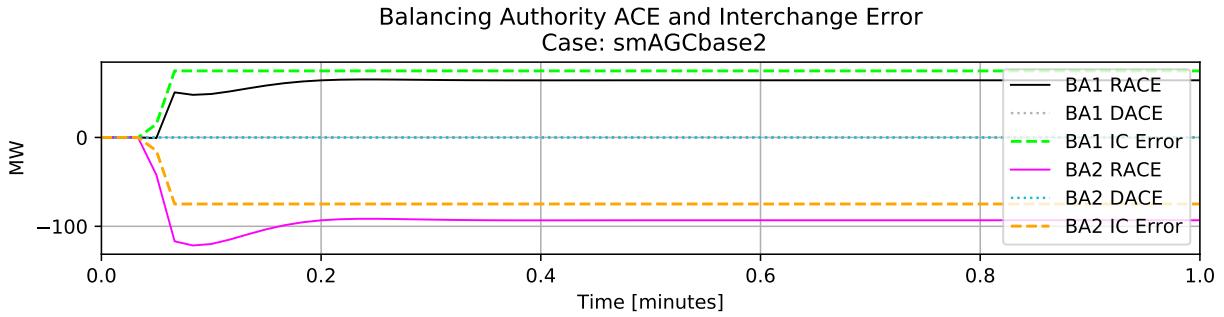


Figure E.2: Calculated values during generation loss event in area 2.

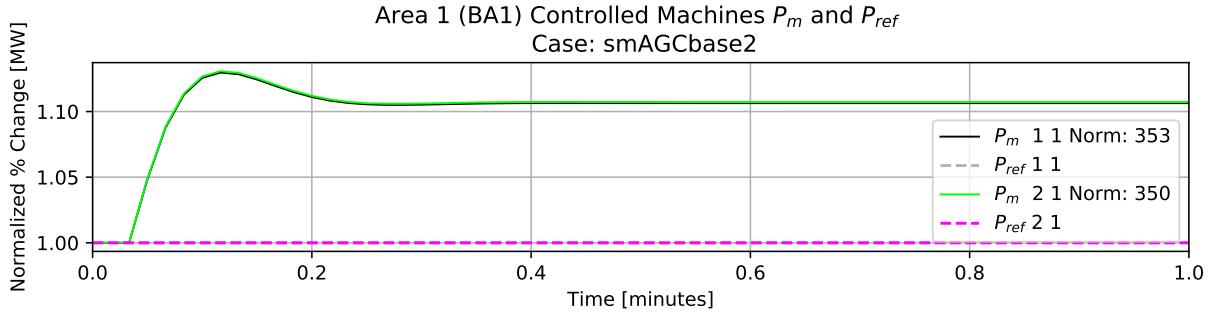


Figure E.3: Area 1 controlled generation response to generation loss event in area 2.

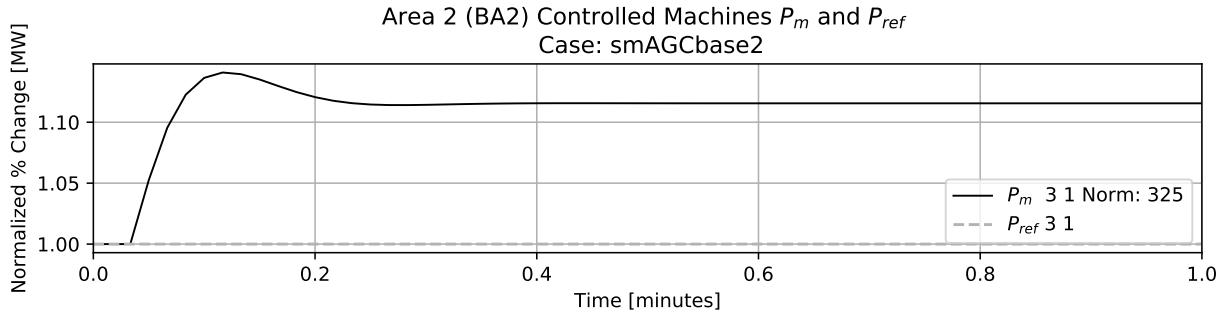


Figure E.4: Area 2 controlled generation response to generation loss event in area 2 .

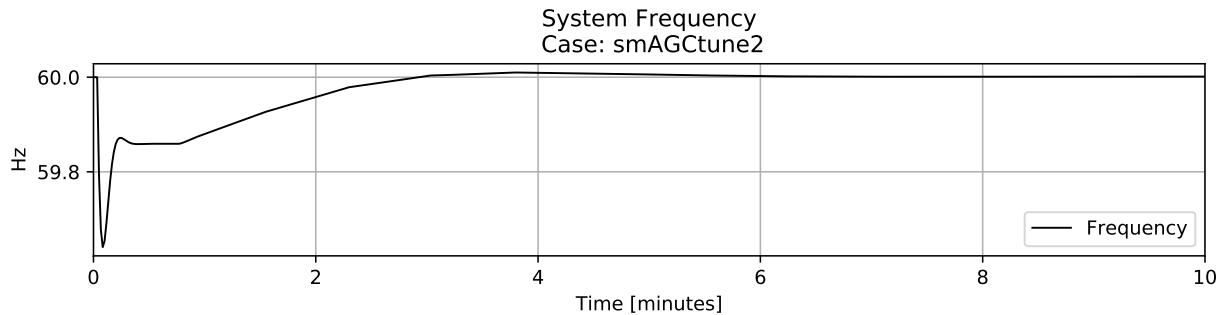


Figure E.5: AGC frequency response to area 2 base case scenario.

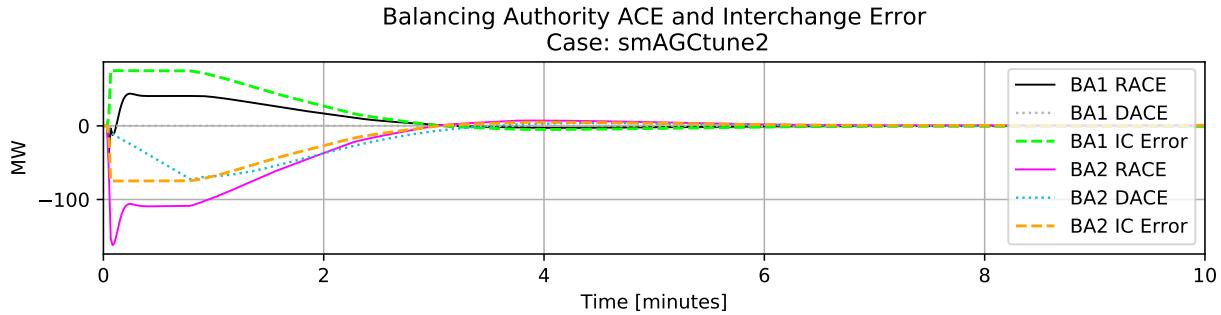


Figure E.6: Calculated BA values during area 2 AGC tuning.

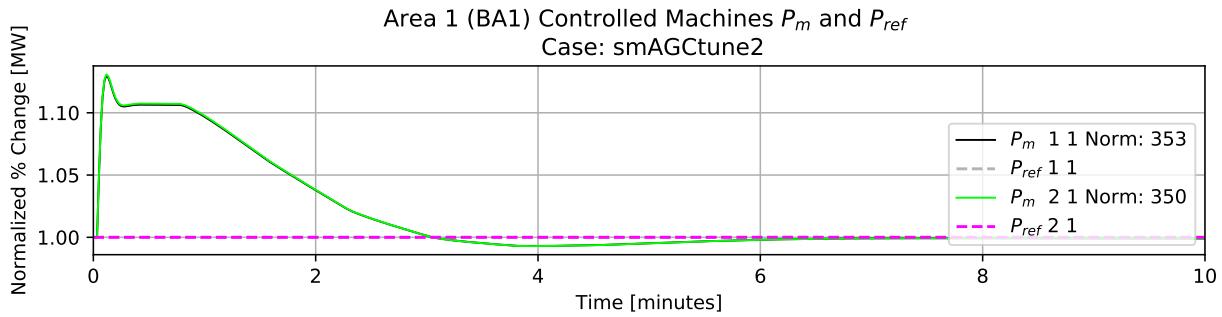


Figure E.7: Area 1 controlled generation response during area 2 AGC tuning.

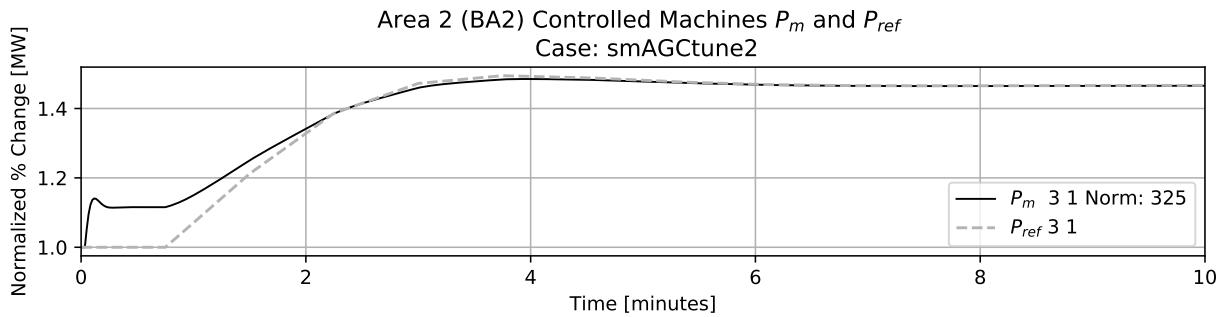


Figure E.8: Area 2 controlled generation response during area 2 AGC tuning.

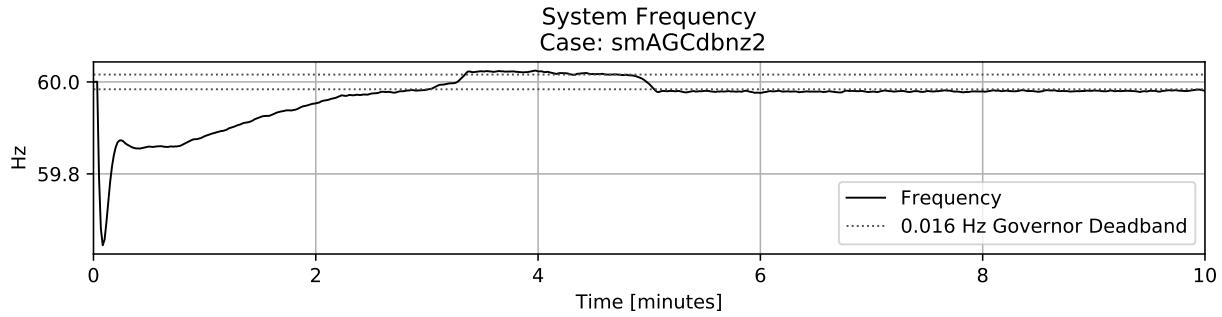


Figure E.9: AGC frequency response with noise and deadbands.

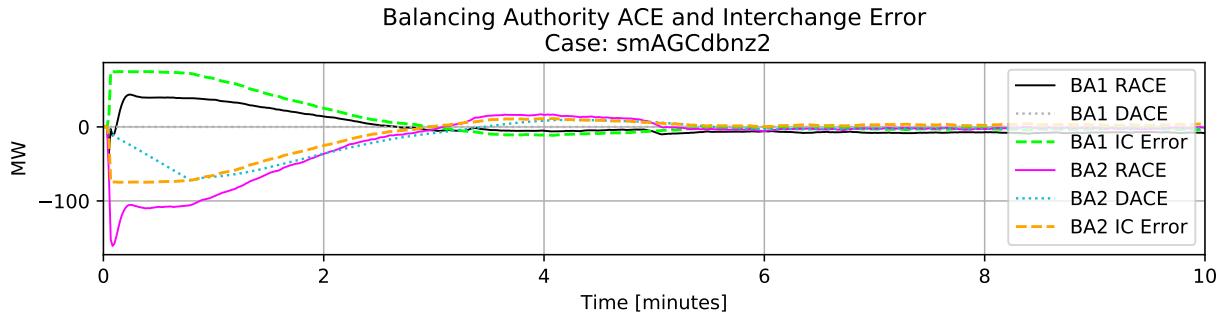


Figure E.10: Calculated BA values with noise and deadbands.

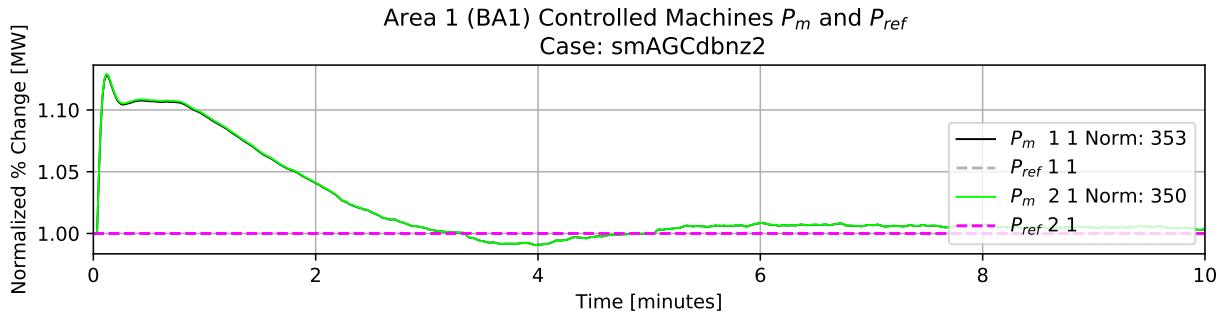


Figure E.11: Area 1 controlled generation response to noise and deadbands.

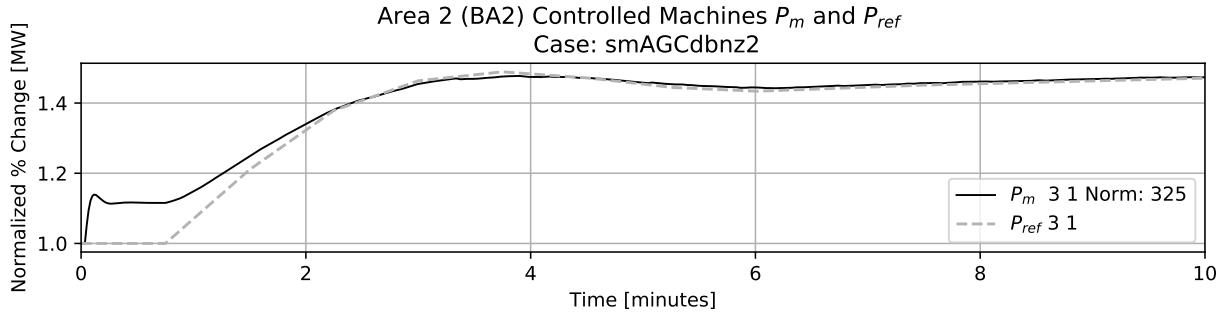


Figure E.12: Area 2 controlled generation response to noise and deadbands.

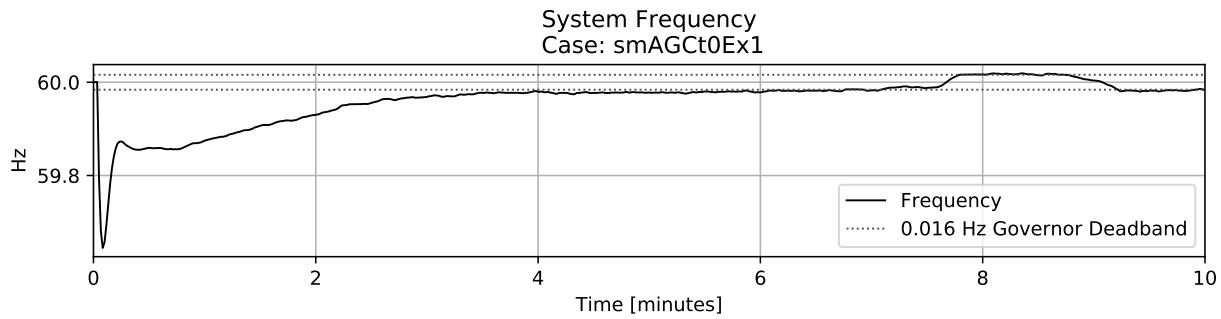


Figure E.13: Frequency response to event using TLB 0.

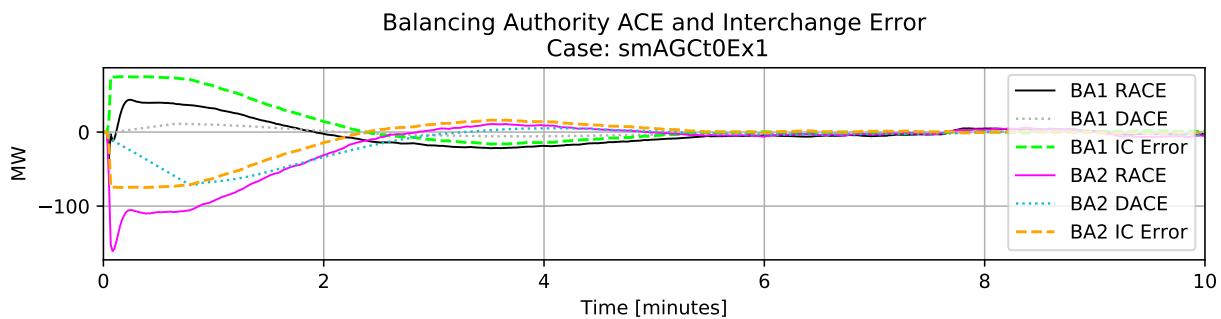


Figure E.14: Calculated BA values during an even using TLB 0.

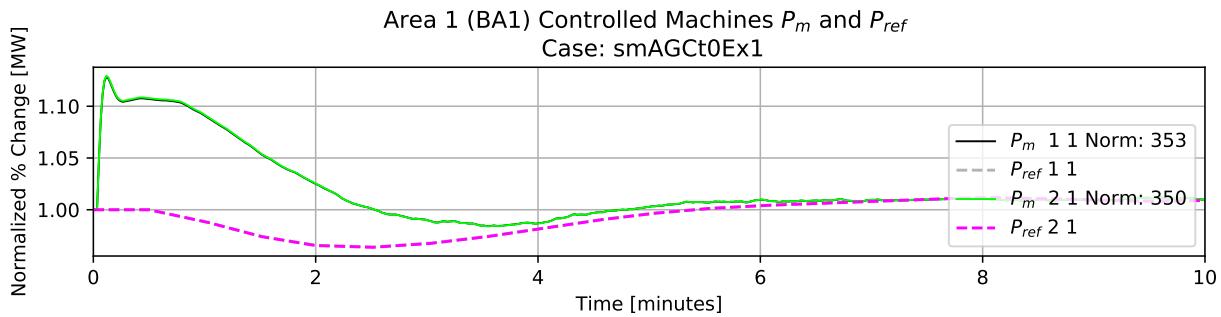


Figure E.15: Area 1 controlled generation response to external area event using TLB 0.

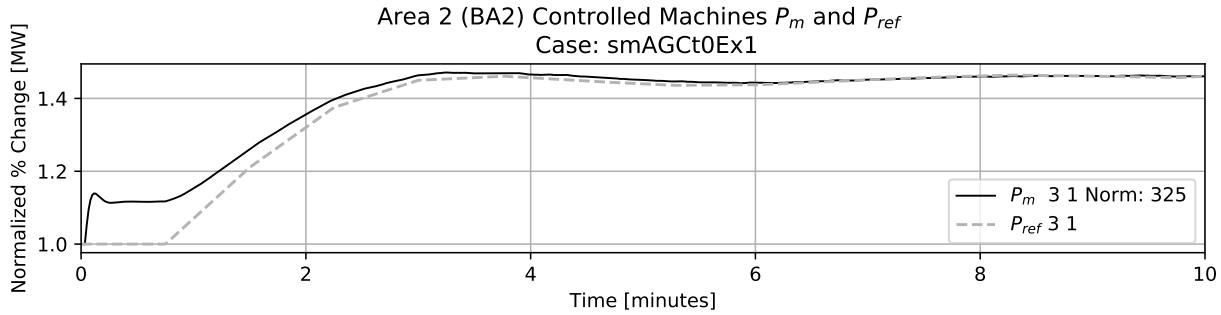


Figure E.16: Area 2 controlled generation response to internal area event using TLB 0.

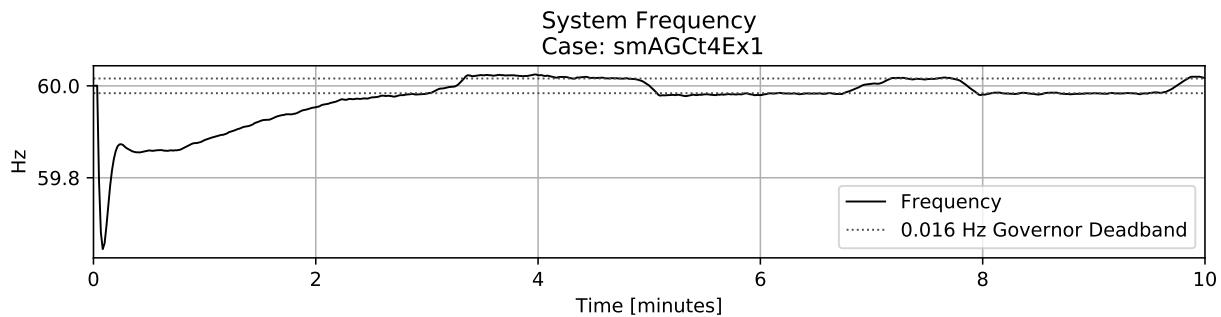


Figure E.17: Frequency response to event using TLB 4.

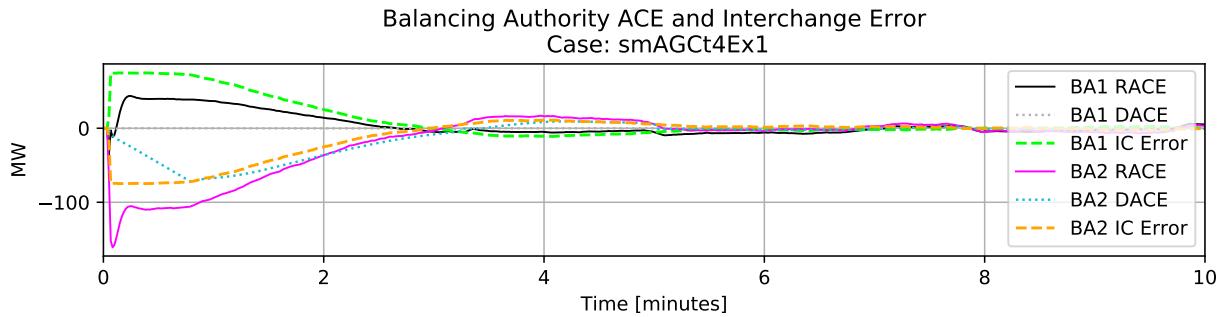


Figure E.18: Calculated BA values during an event using TLB 4.

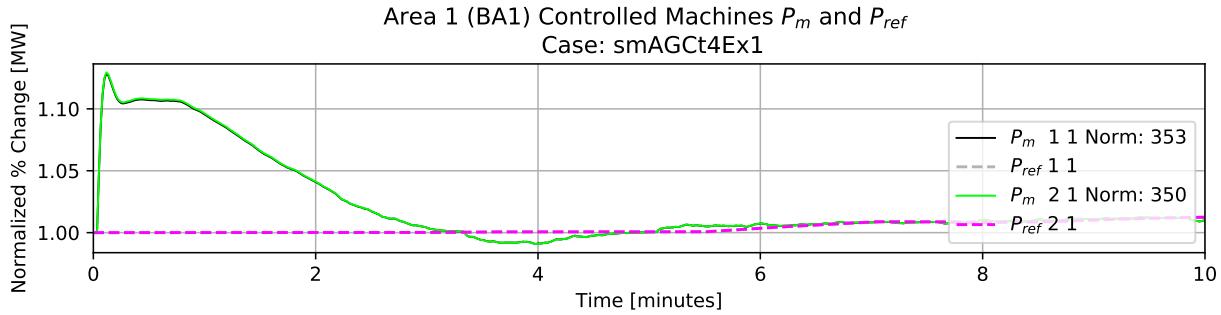


Figure E.19: Area 1 controlled generation response to external area event using TLB 4.

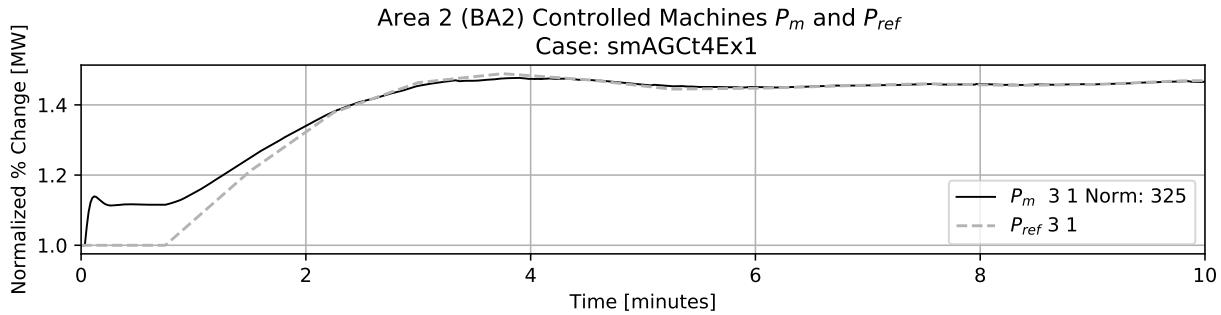


Figure E.20: Area 2 controlled generation response to internal area event using TLB 4.