GE Global Research

Fast Dynamic Simulation-Based Small Signal Stability Assessment and Control

DoE Award #: DE-OE0000626

Final Technical Report

Date: 3/31/2015

Project Lead:

GE Global Research

Naresh Acharya

Principal Investigator

Project Partners:

GE Energy Consulting

Pacific Northwest National Laboratory

Southern California Edison

imagination at work

Acknowledgement

"This material is based upon work supported by the Department of Energy under Award Number DE-OE0000626"

The authors would like to thank Mr. Keith A. Dodrill and Mr. Gil Bindewald with the U.S. Department of Energy for their support and productive discussions during the course of the project. The authors would also like to thank Dr. Nagy Abed and Dr. Farrokh Habbi-Ashrafi with Southern California Edison for sharing their practical experiences and providing insightful comments.

Disclaimer

Primary Authors

GE Global Research Team:

Dr. Naresh Acharya

Dr. Chaitanya Baone

Dr. Santosh Veda

Dr. Jing Dai

Dr. Nilanjan Ray Chaudhuri
(currently with North Dakota State University)

GE Energy Consulting Team:

Dr. Bruno Leonardi

Dr. Juan J. Sanchez-Gasca

Pacific Northwest National Laboratory Team:

Dr. Ruisheng Diao

Dr. Di Wu

Dr. Zhenyu (Henry) Huang

Dr. Yu Zhang

Dr. Shuangshuang Jin

Dr. Bin Zheng

Dr. Yousu Chen

Executive Summary

Power grid planning and operation decisions are made based on simulation of the dynamic behavior of the system. Enabling substantial energy savings while increasing the reliability of the aging North American power grid through improved utilization of existing transmission assets hinges on the adoption of wide-area measurement systems (WAMS) for power system stabilization. However, adoption of WAMS alone will not suffice if the power system is to reach its full entitlement in stability and reliability. It is necessary to enhance predictability with "faster than real-time" dynamic simulations that will enable the dynamic stability margins, proactive real-time control, and improve grid resiliency to fast time-scale phenomena such as cascading network failures. Present-day dynamic simulations are performed only during offline planning studies, considering only worst case conditions such as summer peak, winter peak days, etc. With widespread deployment of renewable generation, controllable loads, energy storage devices and plug-in hybrid electric vehicles expected in the near future and greater integration of cyber infrastructure (communications, computation and control), monitoring and controlling the dynamic performance of the grid in real-time would become increasingly important. The state-of-the-art dynamic simulation tools have limited computational speed and are not suitable for real-time applications, given the large set of contingency conditions to be evaluated. These tools are optimized for best performance of single-processor computers, but the simulation is still several times slower than real-time due to its computational complexity. With recent significant advances in numerical methods and computational hardware, the expectations have been rising towards more efficient and faster techniques to be implemented in power system simulators. This is a natural expectation, given that the core solution algorithms of most commercial simulators were developed decades ago, when High Performance Computing (HPC) resources were not commonly available.

This project team, led by GE Global Research (GEGR), in collaboration with GE Energy Consulting (GEEC), Pacific Northwest National Laboratory (PNNL) and Southern California Edison (SCE) has formulated and implemented a dynamic simulation method that exploits HPC techniques to enable power system dynamic simulation based analysis and control closer to real-time, as opposed to the traditional approach of performing such simulations in offline studies. The project team has demonstrated speed improvements using modified solution architecture on GE's Positive Sequence Load Flow (PSLF) dynamic simulation software. Furthermore, the team has developed a novel Fast Contingency Screening And Control Action Engine (FCSCAE) that can enable real-time small signal stability assessment and control at the power system control center.

The following are the key project results:

1. A novel time stacking method was formulated and implemented for speeding up power system dynamic simulation using high performance computing (HPC). In this method, differential equations in dynamic simulation are algebrized for several time steps. The resulted algebraic equations representing multi-step power system dynamics are then solved using Newton's method. Unlike conventional simultaneous methods where differential-algebraic equations (DAEs) are solved one step after another, the time stacking method is able to solve DAEs over multiple time steps simultaneously. The team has implemented and tested the time stacking

method on a multi-core and shared-memory HPC machine. It is found that time stacking method has the potential to accelerate dynamic simulation by efficiently utilizing more computing capability of HPC.

2. A significant speed improvement of the dynamic simulation core of GE's Positive Sequence Load Flow (PSLF) software was achieved. The solution architecture of GE PSLF dynamic simulation engine and how it can be improved were the major focus of this study. Speed improvements were achieved by parallelizing and updating core solution algorithms of the program. Speed ups greater than 2x have been observed in the parallel version of PSLF. Results have been confirmed and validated by testing the parallel version of PSLF on PNNL's super computers and preliminary results indicate that the solution accuracy remains unchanged.
3. A novel FCSCAE has been developed and demonstrated to enable small signal stability assessment and control in real-time. An eigenvalue sensitivity based contingency screening and ranking algorithm was developed to screen the long list of possible contingencies, and arrive at a small subset of contingencies. This tool ensures that when analyzing large scale power systems with thousands of nodes/ buses, the contingency analysis for small signal stability assessment can be performed at a fast rate. Once critical contingencies are identified by this tool, each of the contingencies can further by analyzed using the detailed time-domain simulation. Furthermore, a novel eigenvalue sensitivity based generator re-dispatch and/or load control algorithm was developed with the goal of improving the small signal stability of the system for not only the current operating condition but also for the critical contingency conditions. The control algorithm leverages the results from the contingency screening tool in the form of eigenvalue sensitivities to identify the most impactful generators and groups of loads that can influence the small signal stability of the system.

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Objectives

Enabling substantial energy savings while increasing the reliability of the aging North American power grid through the improved utilization of existing transmission assets hinges on the adoption of wide-area measurement systems (WAMS) for power system stabilization. However, the adoption of WAMS alone will not suffice the power system to reach its full entitlement in stability and reliability. The need is to enhance predictability with "faster than real-time" dynamic simulations that will enable dynamic stability margin, proactive real-time control and improve grid resiliency to fast time scale phenomena like cascading network failure.

GE Global Research (GEGR) in collaboration with GE Energy Consulting (GEEC), Pacific Northwest National Laboratory (PNNL), and Southern California Edison (SCE) formed a team to leverage scientific advancements in mathematics and computation for application to power system models and software tools. This team was tasked to develop and apply advanced computational techniques to enhance the speed of the dynamic simulation software. This was complemented with GEGR's expertise in small signal stability to develop a proof-of-concept for fast contingency screening and control action engine (FCSCAE). The FCSCAE is targeted to be an Energy Management System (EMS) application with three key elements 1) Fast Dynamic Simulation, 2) Contingency Screening and Ranking, and 3) Control Action Engine. The technologies developed will enable fast, high fidelity capabilities that improve grid reliability in a large scale, dynamic environment.

Specific objectives include:

- Develop mathematical and high performance computing (HPC) techniques applicable to power system fast dynamic simulation.
- Implement high performance computing techniques in power system dynamic simulation software.
- Develop fast contingency screening and control action method for small signal stability.
- Verify and validate speed enhancement of dynamic simulation and decision making methods.

1.2 Background

Since the 1996 blackout in the Western Electricity Coordinating Council (WECC) system, there has been renewed emphasis on continuous measurement-based system monitoring to avoid or restrict the spread of such a collapse [1]. Several events of abrupt line-tripping, load, and generation shedding took place during this break-up, leading to changes in modal behavior of the system [2], [3]. The cascade failure could possibly have been avoided through appropriate operator intervention with accurate and real-time knowledge of system frequency and damping, which are vital indicators of system stress and stability [4], [5].

To date, utilities have considered using the WAMS technology for situational awareness by the operators—which is essentially a monitoring application. As a part of the North American

Synchrophasor Initiative (NASPI), many PMUs have already been installed in the US power grid [6]. Researchers at PNNL, Montana Tech, and the University of Wyoming have developed a grid oscillation detection system called "Mode Meter" [4] which uses real-time data combined with new analytic methods, and reports to operators with an easy-to-read visualization tool. Many universities, vendors, and utilities have combined knowledge over time to develop such applications using different signal processing techniques [5], [7], [8], [9] and these are being used today in control centers [10], [11], [12].

Even after equipping the grid with PMUs and situational awareness tools, a cascaded blackout took place in the Arizona-Southern California region on September 8, 2011. A detailed investigation report [13] by the Federal Energy Regulatory Commission/North American Electric Reliability Corporation (FERC/NERC) indicated lack of adequate real-time situational awareness of conditions and contingencies throughout the Western Interconnection. The executive summary reflected: "...many entities' real-time tools, such as State Estimator and Real-Time Contingency Analysis (RTCA), are restricted by models that do not accurately or fully reflect facilities and operations of external systems to ensure operation of the Bulk Power System (BPS) in a secure N-1 state. ....The lack of adequate situational awareness limits entities' ability to identify and plan for the next most critical contingency to prevent instability, uncontrolled separation, or cascading outages. If some of the affected entities had been aware of real-time external conditions and run (or reviewed) studies on the conditions prior to the onset of the event, they would have been better prepared for the impacts when the event started and may have avoided the cascading that occurred."

Present day dynamic simulations are done only during offline planning studies. The state-of-the-art dynamic simulation tools have limited computational speed and are not suitable for real-time applications, given a large set of contingencies to be evaluated. These tools are optimized for the best performance of single-processor computers, but the simulation is still several times slower than real time due to its computational complexity. With the latest development in numerical methods and computational hardware, there has been the expectation that more efficient and faster techniques are implemented in power system simulators. This is a natural expectation given that the core solution algorithms of most commercial simulators were developed decades ago, when HPC resources were not commonly available.

Recently, there have been an increasing number of efforts aimed at increasing the computational speed of the power system dynamic simulation. Researchers at PNNL have shown that the traditional power system computation should be reformulated to take advantage of the high-performance computing platform. It has also been indicated that with parallelization and significant speed-up, it is possible to achieve a faster than real-time dynamic simulation.

In addition, a part of the computation burden can be relieved by developing a fast contingency screening and ranking tool. The advantage of fast dynamic simulation can then be taken to develop an application to guide system operators in real-time to take definitive control action.

In summary, the team proposes to develop and implement high performance computational (HPC) techniques such as the simultaneous method and preconditioned conjugate gradient method to accelerate the speed of the power system dynamic simulations. These HPC capabilities combined

with screening and prioritizing contingencies are proposed for proactive decision making (control) for system operator to improve small signal stability. GE's existing PSLF simulation software gives us an excellent platform to commercialize the resulting technologies.

1.3 Organization of the report

This report is organized into eight sections. The key objectives of the project along with the brief background are described in first section. The second section outlines the key challenges associated with the enhancing the speed of the dynamic simulation and some potential approaches. The third section describes in detail the steps taken to improve the simulation speed of the PSLF dynamic simulation engine along with the lessons learned during the process. A simultaneous time stacking method which has the potential to significantly improve the speed of dynamic simulation is formulated in fourth section of the report. A fast contingency screening and oscillation damping control action method is described in section five and six respectively. Section seven lists the relevant references. Lastly, the publications and patents that best describes the outcome of the project is listed in section eight.

2 Mathematical and computational techniques for fast dynamic simulation

Assessing power system transient stability under contingency conditions via dynamic simulation is of critical importance to the secure operation and planning of a power system. It solves a set of differential-algebraic equations (DAEs) many times during one simulation that describes the electro-mechanical interaction of generators as well as power electronic devices and their controllers. Due to the extensive computational requirement and increasing model complexity, it is several times slower than real time to run a single time-domain simulation for a large-scale system model, such as WECC or Eastern Interconnection in North America. Currently, most of the commercial software tools for dynamic simulation use explicit integration methods to solve the differential-algebraic equations. Due to the numerical instability issues inherent with explicit methods, a small time step, e.g., ¼ cycle or ½ cycle, is typically required to ensure simulation accuracy and numerical stability in estimating the dynamics of the power system. The small time step can be a fundamental challenge that affects the speedup of dynamic simulations.

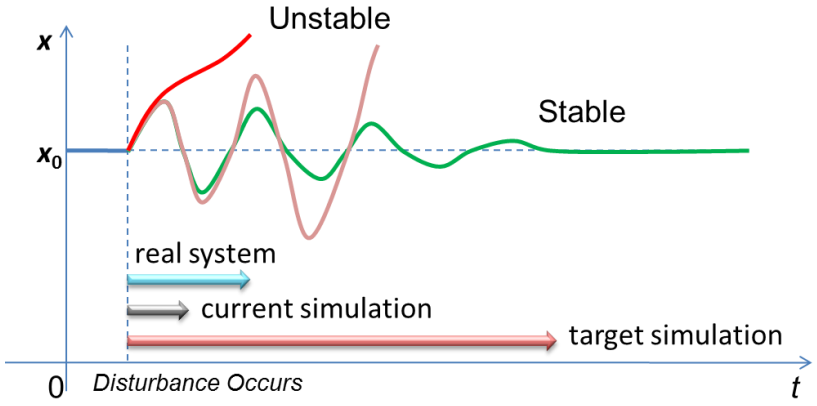


Figure 1. Obtaining time-domain trajectory using dynamic simulations

There are several ways to improve power system dynamic simulation speed. One popular way is to reduce system model complexity so that the total amount of computation time can be reduced. There is no doubt that the computation time can be significantly reduced by model reduction. However, detailed behavior of power system can be easily lost so that the simulation accuracy cannot be guaranteed. Another method is to use distributed simulation for massive contingency analysis. This method is designed to distribute a large number of simulation cases to different cores/computers so that the total amount of simulation time can be reduced, but this approach cannot speed up each individual simulation case. For a large-scale power grid, the simulation speed cannot reach real-time requirement to apply control action in time. It is very challenging to further reduce the simulation time without compromising simulation accuracy by varying time steps, model reduction and/or other techniques. The main reason is that today's commercial tools for dynamic simulation (e.g., PSLF by GE, PSS/E by Siemens PTI, and TSAT by Powertech Labs) are designed, optimized and operated on a single-core machine, while the main frequency of CPUs is not increasing because of physical constraints. Speeding up dynamic simulations via parallel computing has been proved to be a feasible and low-cost solution. A variety of parallel algorithms were

proposed decades ago, including parallel in time, parallel in space, and waveform relaxation. However, very few algorithms have been actually adopted by major software vendors, because: (1) fast parallel computers with reasonable costs were not available back then; and (2) it requires a significant amount of time and efforts to revise the source code of those stability programs that were developed ~30 years ago.

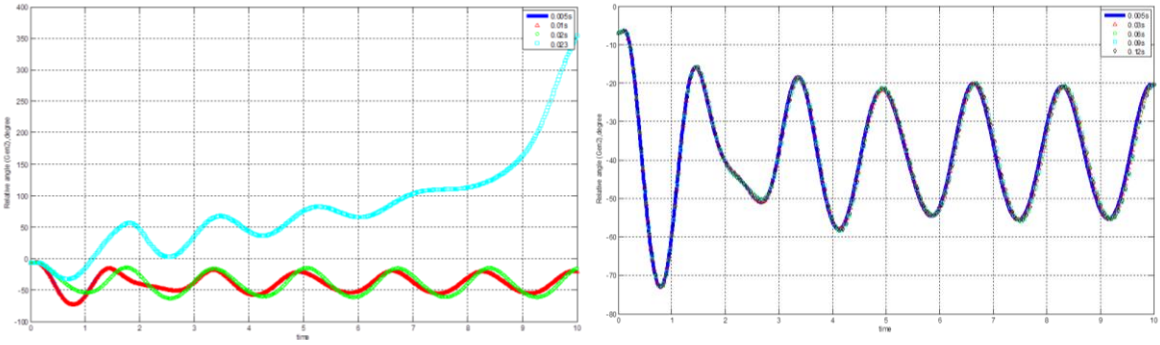


Figure 2. Implicit integration shows better numerical stability with larger time steps, tested on a 2-area system (Left figure: modified euler method; Right figure: Trapezoidal integration method)

In this research effort, several methods are explored in order to improve the speed of a single dynamic simulation. The first way is to leverage ongoing research work performed at PNNL to apply HPC techniques to the explicit integration methods. The main idea is that for each time step marching, differential equations representing generator dynamics are allocated to different cores of HPC platform because the generator dynamics can be computed independently. Regarding solving the network algebra equation, an efficient linear solver is required for speeding up dynamic simulation. Such innovative ideas can be directly integrated into PSLF. Another promising method to further speed up power system dynamic simulation is to reformulate the problem so that the differential equations and network algebra equations are solved simultaneously at each time step. In this method, the state variables at the current step are modeled as a function of both the values at previous steps and themselves. To solve such problem requires implicit integration methods, which have better numerical stability compared to explicit methods. That means, a larger time step can be used for improving simulation speed. Our literature review shows that both the explicit methods and implicit methods were proposed for power system simulator decades ago. However, the performance of implicit methods implemented on a single-core computer was not superior to explicit methods, according to a test conducted by GE about two decades ago, due to the following reasons:

1. It can take much time to compute the Jacobian matrix at each time step if using single-core computer.
2. Iterative methods are needed to solve for the system equation in the implicit form. Convergence can be a limiting factor without a good linear solver.
3. The algorithms were written to run on single-core computers. The HPC hardware platform was not ready for implementing more advanced algorithms.

With the development of HPC techniques and advanced algorithms for linear solvers for the past decade, these bottlenecks can be easily overcome, given the fact that PNNL has already built up such capabilities, in that: (1) computing the elements of the Jacobian matrix can be allocated to different processors at the same time to save time; (2) PNNL has implemented advanced iterative solver that has good scalability with good convergence performance, which will be introduced in the section below; and (3) the HPC platform becomes much more affordable with mature algorithms. With such advantages, the implicit integration method can be of great potential to speed up dynamic simulation so that faster-than-real-time simulation capability can be achieved for a large-scale power system without model reduction. Unlike other published work on HPC implementation in dynamic simulation that is performed on simple power system dynamic models, the collaboration of GE and PNNL makes it possible the advanced algorithms developed in this project be directly integrated into the commercial software for industry members to use, because PSLF has the entire model library of commonly used power system devices.

Initially, GE Energy Consulting (GE) and Pacific Northwest National Lab (PNNL) have been tasked with developing a parallel architecture of a power system simulator that was faster than real time. Because a major modification of the PSLF dynamic solution engine is infeasible due to time and resources constraints set in this project, two different paths were initially considered in order to improve chances of a successful project completion. It is worth mentioning that although each team had well-defined and independent tasks to perform, they interacted heavily and a substantial amount of knowledge was exchanged in the course of the project.

The two proposed approaches to develop a parallel version of a power system simulator are described below:

- **Approach 1:** Leverage current PSLF solution architecture and use APIs for multiprocessing computation (such as MPI/Open MP) recent advances in linear system solvers to improve simulation speed.
- **Approach 2:** Develop an alternative solution architecture of the system of differential-algebraic equations (DAE) used to represent the electric power system and its components. Such architecture would be multicore-friendly to but would require a significant effort and major re-writing PSLF's solution engine.

Advantages and disadvantages of each approach are summarized in Table 1.

Table 1: Summary of advantages and disadvantages of parallelization approaches

Approach 1	Approach 2
PROS • Faster to implement in PSLF since the solution architecture of the program does not need to be completely re-written • Less code changes require less effort to translate the prototype into an eventual commercial version of the tool	**PROS** • Expected to provide greater speed gains than approach 1, especially if system of DAE equations can be made very large to leverage multiprocessing more efficiently • Can be used as the basis for the development of next generation of tools and additional tools, such as a small signal analysis tool.
CONS • Speed gains are limited by the speed of slowest loop on current program architecture – the speed bottleneck • This approach still requires significant changes that can be challenging and require a significant amount of resources to be performed	**CONS** • More difficult to be implemented and be translated into a commercial grade application • More code changes and more code development • Would require a significant effort to the current solution architecture in PSLF

After a team review and considering the resources available in this research project, it has been determined that approach 1 would be carried out by the GE team, with the PNNL team being responsible for implementing approach 2. Section 3 describes the effort related to approach 1 conducted by GE team on improving the speed of PSLF simulation engine. Section 4 describes the alternative simulation architecture developed by PNNL to significantly improve the speed of dynamic simulation.

Such task division is aimed at trying to maximize the project success by executing two parallelization strategies concurrently. Nonetheless, both GE and PNNL team have exchanged a large amount of experience, information and knowledge in this project, which was an important aspect in its successful completion.

3 Parallelization of existing PSLF solution architecture

The first step to improve the solution speed of PSLF is to investigate where the program spends most of the time in the course of a dynamic simulation. A thorough review of the solution algorithm was performed in this study to identify potentially feasible steps that could be taken in order to parallelize the program architecture.

Several different attempts have been made before achieving a successful parallel architecture described in this report. Because many of these attempts were unsuccessful at improving solution speed, significant insight was gained in the process by better understanding how the dynamic simulation of power systems could be effectively formulated to explore latest advancements in solver and parallel architectures. Despite several failed attempts, a combination of certain combination of improvements has been shown to make the dynamic simulation of large electric power systems faster. The major enables of such speed improvements are: updated ordering algorithms, updated linear solvers, extensive usage of sparse matrix manipulations and the use of multiple threads by the solution engine.

Parallelizing the solution of power system dynamic simulations involves the solving a system of algebraic and differential equations in parallel. Because PSLF relies on particular solution architecture, both the solution of algebraic and differential equations needs to be parallelized or accelerated. Failure in doing so will cause the non-improved portion of the core to limit parallelization gains obtained due to use of multithreaded computations.

3.1 PSLF solution architecture

The GE PSLF program is a well-accepted software solution in the electric power industry. The program can analyze the electric power system in three major areas, namely: power flow and steady state, short circuit and stability. This study is focused at accelerating the portion of the program that performs the stability analysis, also frequently referred to as dynamic simulation. During the course of this report, the term dynamic simulation will be used to refer to the solution of the differential and algebraic equations used to model the system.

The first step in accelerating the program execution is a careful evaluation of the current program architecture and identification of bottlenecks were much of the simulation time is spent. Such analysis may seem trivial but given the extension of the program source code, such step can be time consuming. Moreover, since the program has been developed over many years, portions of the code lack proper documentation, making the process of understanding program operation a more challenging.

The PSLF dynamic solution architecture can be divided in to two major parts: the network solution and the model solution. Network solution is concerned with the solution of the network equations given the current injections at the boundary of the network. These current injections are obtained from the solution of the models that are connected to the network. In a broad sense, the model being solved can be represented by equation

$$0 = \mathrm{I} - YV$$
$$\dot{\mathrm{x}} = \mathrm{f}(\mathrm{x}, \mathrm{V}) \qquad (1)$$

In (1), algebraic variables are represented by in the array **V** and include bus voltage magnitudes and angles. The current injection at the system buses is represented by the array **I**, whereas array **x** is used to represent all states variables present in the dynamic models used in a particular simulation.

Since it would be a significant effort to describe all possible state variables used in a simulation, the user is referred to the PSLF user manual for as additional details regarding set of differential variables used on each dynamic model.

Traditionally, the solution of (1) is generally accomplished via two major approaches: a partitioned solution or simultaneous solution.

The partitioned solution solves the algebraic equations separately from the differential equations, thus the name *partitioned*. Initially, the program is solved for a steady state solution of the network, representing the current power flow at a specific operating condition. The solution of the algebraic equations (network solution) is often accomplished with a sparsity oriented factorization method, followed by a forward and a backward sweep. Differential equations are solved in multiple ways and by different integration algorithms. Popular methods include both explicit algorithms such as Adams-Bashforth family (Euler method is one of the most popular variants of the Adams-Bashforth integration algorithms), and implicit methods, such as the trapezoidal, Krylov subspace methods such as the GMRES, BCG and implicit Runge-Kutta algorithms. For a more detailed discussion of different integration methods, the reader is directed to [14], [15].

Several intricacies of the dynamic models play a major role in determining which integration method should be used to solve the differential system of equations. The stiffness of the system of equations, which is characterized by a ratio between the largest and smallest system time constant, is critical in determining the algorithm used and an adequate integration step size. The number of steps used in the integration method, i.e., the numbed of past time steps necessary to calculate the current integration time step, also affects the integration step size, stability and accumulated error of the methods.

PSLF uses a partitioned solution scheme with an explicit integration method and constant integration time step. This is a very popular method amongst commercial grade power system simulators that perform dynamic simulation. This is also the method of choice in many programs that perform simulations in the electromagnetic transient realm, such as the Electromagnetic Transient Program (EMTP) developed by BPA. On the other hand, there are commercial tools that use implicit integration methods combined with variable integration time step.

In the PSLF architecture, the network voltages are initially solved for using a power flow solution. The initial voltage values are then passed to the models connected to the network, which will then calculate how much current injection shall be produced by them. Once the nodal current injections are found, they are passed to the network model and the first equation shown in (1) is solved for **V.** This process is repeated until the end of the simulation. Figure 3 shows a sketch of how the

partitioned method exchanges the necessary information across an interface while solving both algebraic and differential equations in a simulation run.

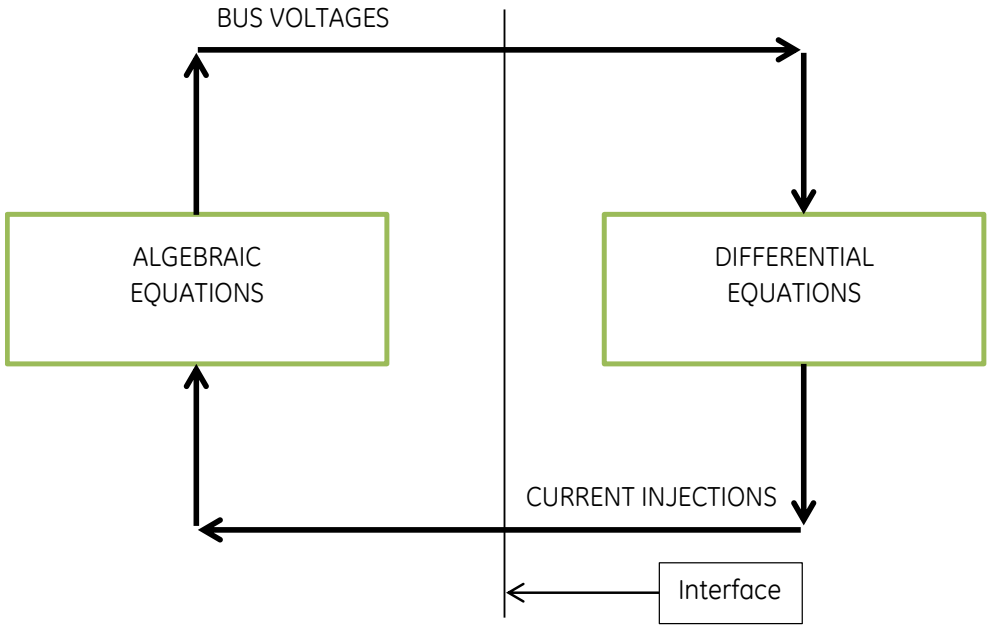


Figure 3: PSLF solution architecture depicting the partitioned solution scheme where algebraic equations are solved separately from differential equations

The flowchart shown in Figure 4 depicts the steps taken in order to solve a dynamic simulation in PSLF. Keywords show in capitalized text helps orchestrate the program execution and are used in the internal structure of all PSLF models, including the user defined ones.

In a dynamic simulation, the user calculated the network solution that defines the actual state of the power flows and voltage levels at all buses in the system. On all voltages are known, all dynamic models need to be initialized based on the current network solution.

The INIT block performs the initialization of all state variables and its derivatives in order to prepare the models for integration during the run.

The SORC step calculates the amount of injected current into the network and is only necessary in models that interface the network, such as generators, motors, SVCs, etc. For instance, electrical controllers and turbine governor systems would not have a SORC block of code in its structure because they do not interface the network directly, but rather connect to a generator model.

NETW calculates the network solution based on injected currents calculated in the SORC block and will then determine the new voltage magnitudes to be used in the next step.

ALGE is often responsible for the calculation of algebraic variables during the simulation process, such as variable wind up limits, integrator limits and any variable of interested that needs to be recorded for later analysis.

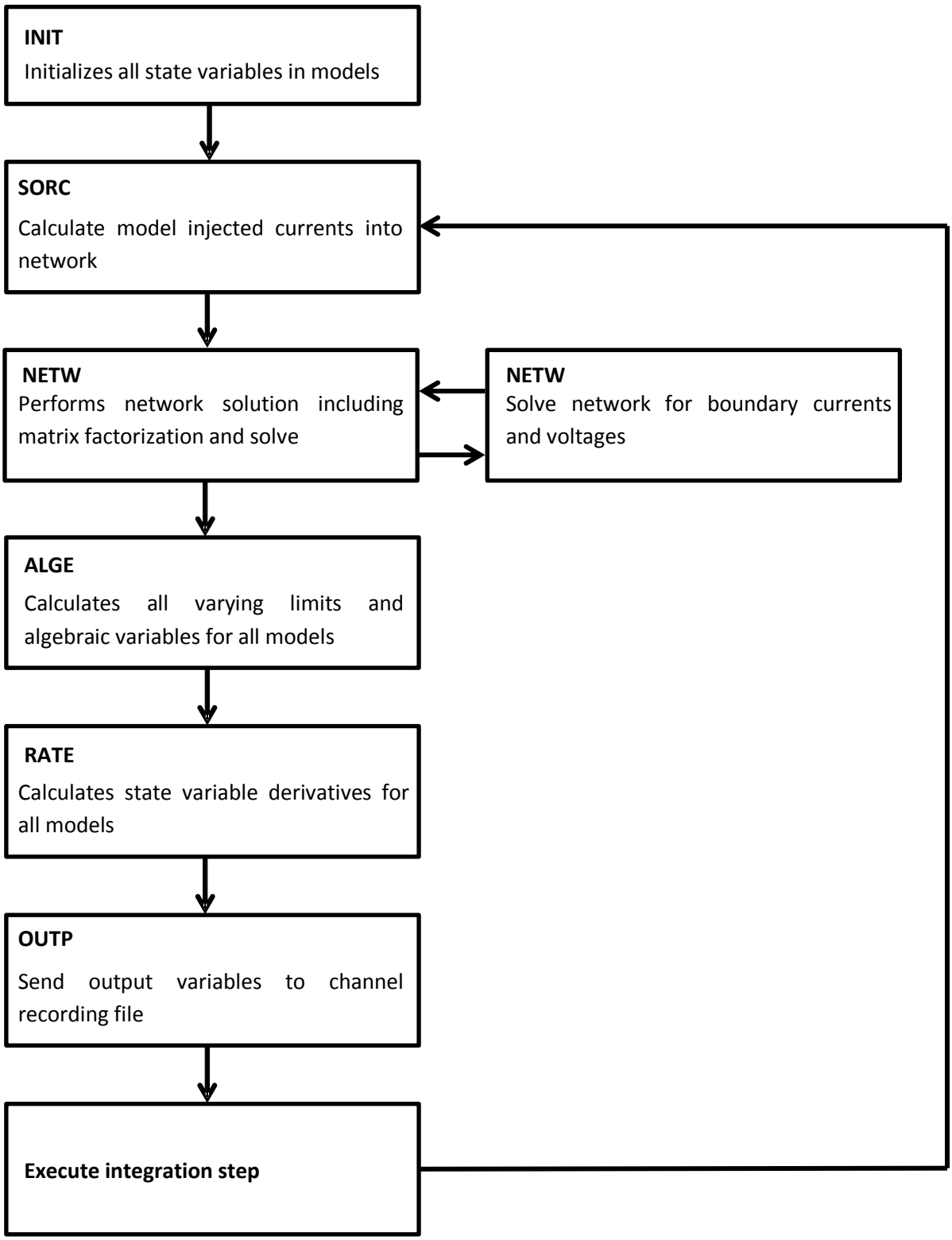


Figure 4: Detailed solution flowchart of PSLF indicating all steps required in order to solve a dynamic simulation

The RATE block performs the calculation of all state derivatives and other necessary variables that are needed in order to perform an integration time step. All models that have states will have to have a RATE block represented in their structure.

All variables recorder for later analysis are specified in the OUTP block. This includes both algebraic and state variables of interest, block diagram outputs and any other value of interest. After all blocks have been executed, the program executes an integration time step and advances the simulation in time. The simulation continues in loop until the end or until the user stops it to introduce a disturbance or a switching event.

3.2 Parallelization of PSLF differential equations with OpenMP

The solution of differential equations accounts for a significant amount of time spent at the PSLF core during a dynamic simulation. Therefore, improving the speed at which these equations are solved is a key step towards achieving a faster simulation engine.

OpenMP is an application program interface (API) that supports multi-platform shared memory multiprocessing programming in C, C++ and FORTRAN, on most processor architectures and operating systems, including Solaris, AIX, HP-UX, Mac OSX and Windows platforms. Its extensive set of directive-based APIs allows for easy and flexible parallelization of earlier, serial programs written in the languages described above.

OpenMP is extensively used to parallelize programs that were originally written to run in serial, with a good flexibility regarding the platforms where it can be used. Some of the latest versions of OpenMP are already integrated in developing environments such as Microsoft® visual Studio 2010, thus making its applicability to existing applications even easier.

A transient stability simulation in PSLF (and other commercial grade simulators) sequentially executes several specific activities at each time step. An integration step is then performed and time is advanced. This cycle will be repeated until the user specified end time of the simulation is reached.

The solution of the differential equations is mainly performed by the activities SORC, NETW, ALGE, and RATE. Two other activities, DYSL and OUTP, account for the solution of the algebraic equations (network solution) and the output of selected variables, respectively. An integration step is then performed by activity STEP after the other activities have been executed at least once and the cycle is repeated until the end of the simulation.

The program structure is serial, thus becoming a good candidate for parallelization with OpenMP. After the parallelization, it is expected that the parallel version of PSLF will be able to share its computational load during a dynamic simulation among several cores and threads. Therefore, only parallelization in space is being attempted in approach 1.

The PNNL team is experimenting with a time-stacking method that performs the parallelization in time along with a parallelization in space. As previously described, a significant effort would be required to implement such method under the existing PSLF architecture and was deemed inviable in this project. The GE team has determined that it would be best to focus its efforts in trying to leverage the existing PSLF solution architecture and extensive model library to the highest extent

possible. On the other hand, the PNNL team has dedicated much of its resources at developing a practical prototype that will step away from conventional solution architectures of power system simulators.

3.2.1 Results of parallelization of differential equation solution

The parallelization of the differential equation solution has been successfully performed for a selected group of generator models and a summary of the results is shown in Table 3.

The test case used to perform these simulations was the Eastern Interconnection and a description of system components is provided in Table 2. For this test, only generator models are switched on and all other models are switched off.

Although only generating units have been considered in this test, these models are detailed and accurately represent the same models used by power system planners and operators daily in the US.

Table 2: Description of test system elements including totals

Element	Count
Buses	70477
Branch Sections	64008
Transformers	21909
Generators	9447
Loads	38196
Shunts	3535
Static VAR devices	6308
DC buses	76
DC lines	38
DC converters	76

The speed gains for a 1s simulation are reported in Table 3 and indicate that some degree of parallelization can be achieved in order to improve performance. Preliminary results indicate that the solution accuracy remains unchanged. When two threads are used, the simulation time is reduced by 15.1% on average. For three threads, it goes up slightly, thus indicating that either saturation started to occur or there are not enough threads available. Nonetheless, a reduction 12.7

% on average is observed when three threads are used. If processes compete for threads, the computer may spend additional time switching between various activities that are competing for a particular core.

Table 3: Parallelization of differential equation solution

Run	Serial	Parallel (2 threads)	Parallel (3 threads)
1	11.10s	9.88s	10.22s
2	11.91s	9.71s	10.15s
3	11.60s	9.78s	10.21s
Average	11.53s	9.79s	10.19s
Gain	-	Reduction of ~15.1%	Reduction of ~12.7%

In order to better understand why saturation was occurring at a relatively low number of threads, a detailed core analysis was performed in order to identify the major bottlenecks of the current PSLF structure. The following list summarizes a couple factors that directly impact the speed of execution of the parallel version of PSLF:

1. **System size:** the simulation of the EI case showed that speed gains could be achieved because the system size was large and a significant large number of computations is required. Consequently, the number of dynamic models and floating point operations (flops) needed to find a solution is high. This is important because the time spent in the blocks of code parallelized by OpenMP need to have enough operations so that the overhead costs of parallelizing can be offset. It has been observed that small test cases with only tens or hundreds of buses and dynamic models, the parallel version of the code may take longer than the serial version due to OpenMP overhead costs being greater than the amount of time required to perform the flops in serial.
2. **Overhead costs of parallelization:** In PSLF and other commercial grade simulators, all dynamic models have their equations arranged internally before the simulation. After the simulation is initialized, the equations are solved at each integration time step. Therefore, the only window where parallelization can be attempted is in between each time step. It is impossible, at least under the current PSLF solution architecture, to perform parallelization of multiple time steps concurrently.

Because most transient stability simulators have a similar structure of PSLF, the parallelization of their architectures will face similar challenges and limitations. One of the most important restrictions is perhaps the fact that a large number of flops per time step are required in order to observe gains in a parallel implementation. Unless a minimal number of computations are executed, the overhead

costs of forking-joining multiple threads cannot be offset. In those circumstances, the serial version of the simulator will be faster than its parallel version.

3.3 Network solution

3.3.1 Identification of fast linear solvers

After parallelizing part of the dynamic engine, a relatively small runtime reduction in the order of 15% was observed. For a more substantial performance improvement and higher speed gains, additional modifications in the PSLF code will be required and that includes a thorough investigation of network solution engine.

During a typical dynamic simulation, the network equations also need to be solved at each time step and during transients. These solutions are typically solved several times at each time step due to the higher rate of change in the system states. Additional solutions are necessary because model current injection on the network boundary buses is constantly changing due the dynamics of the models.

Network solution is composed of three main steps: ordering, factorization and solve. In a typical algorithm, the factorization step accounts for most of the time taken to find a solution (around 95%), while the forward/backward is responsible for the remaining 5%. Factorization is only required when the network topology changes, i.e., during the beginning of the simulation, during switching events and during faults. It is also one of the computationally tasks in a dynamic simulation, even though it is only performed a few times during the course of a solution. Therefore, an improvement on the linear solver used could help expedite the solution, making the overall simulation faster.

A solve needs to be performed after the network matrix is factorized in order to calculate system voltages. Although the solve step only accounts for a small portion of the time involved in the network solution (around 5%), it accounts for a substantial portion of the time spent during the simulation because it is invoked several times over the course of a single dynamic run. Therefore, if the forward/backward solution can also be accelerated, then the overall system solution time could be significantly reduced [16]. Improvements in the factorization and solve routine can provide observable improvements in simulations within typical transient stability time frame (10-30s).

3.3.1.1 Parallelization of solve (forward/backward substitution)

Since most time consumed in the network solution is spent at the solve step, an attempt to parallelize the forward and backward substitution algorithm available in PSLF was made. Despite the fact that the forward and backward algorithm is inherently serial and has little room for effective parallelization, some level of parallelization can be implemented [16]. The idea is to explore the algebraic variables that are already computed and run subsequent calculations in parallel.

The parallel solve method was successfully implemented but did not show significant speed improvements mainly due to the fact that the amount of flops executed in parallel was not enough to justify the overhead costs of parallelization. Therefore, the idea of parallelizing and accelerating the linear system solution in PSLF was dropped altogether and other options needed to be explored.

A possible way to improve the execution speed of the linear system solver involved with the network solution was to replace the native PSLF solver. The rationale was that if an up to date solver with superior execution speed in ordering, factorization and solve could be implemented and used to replace the current PSLF solver, speed gains would be achieved.

3.3.2 Literature review

In order to replace the PSLF solver with a solver that could provide additional speed gains, a detailed literature review was performed to identify a suitable direct method solver that could provide performance improvement. The solver needs to have demonstrated performance on circuit solutions and must also be able to be compiled and linked as a library in the PSLF model. After a few solvers are identified as potential candidates for replacement, the most promising ones will be tested in matrices representing real size networks.

3.3.3 Direct methods and iterative methods

Before deciding which solver could be used to replace the current PSLF solver, a review of current state of the art linear system solvers was performed in order to find a suitable replacement. Since PSLF has its architecture already developed around a direct method, there is a slight preference to replace its current direct method solver with another direct solver. Another factor influencing the choice of a direct method instead of an iterative method is that direct methods tend to be faster than iterative methods for the problems with dimension of order near 100,000.

In the past 50 years, extensive work has been done in the area of linear system solution and comparison has been performed in order to determine the most suitable solvers for each problem at hand [17], [18], [19]. The general consensus is that there is not a single "silver bullet" solver that is capable of outperforming all of its competitors for a wide range of matrices that arise from different mathematical problems. Much of the success of a linear system solver is dependent upon the matrix characteristics of the problem at hand.

For instance, the SuperLU solver and is capable to outperform several other solvers in terms of speed by exploiting the so called supper nodes. Super nodes are a group of continuous factor columns with nested structures that can be solved together. What makes super LU fast is the capability to explore these super nodes. However, not all mathematical problems (including power system networks) have a substantial amount of super nodes that SuperLU could take advantage of. Therefore, other apparently "less sophisticated" methods can outperform SuperLU when applied to power system problems.

The network matrices that arise in power system modeling are often very sparse, quasi-symmetric and lack many characterizes that could be explored by several recent solvers. Therefore, choosing an appropriate solver is key in order to ensure speed gains and must take into account the problem being addressed.

3.3.4 Ordering

Sparsity oriented methods and adequate matrix ordering are critical to achieving a more efficient use of computational resources, maximizing speed and minimizing memory storage needs. Without proper ordering, matrices that originate from modeling the largest US interconnections could not be

efficiently solved in modern computers and without sparse storage methods, it would require significant memory to store these matrices on a typical desktop/laptop computer.

Therefore, finding an adequate ordering method is a key step in improving the entire speed of solution. In this project, several ordering methods were tested and the fastest algorithm was selected to be implemented in PSLF.

The efficacy of the selected algorithm in ordering a typical large size power system matrix can be visualized in Figure 5. The figure on the left hand side shows the WECC network matrix before ordering, whereas the left hand side picture shows the same matrix after running the AMD algorithm. The dimension of the matrix is around 18,000. As expected, the matrix is very symmetric with respect to its main diagonal and exhibits significant sparse characteristics.

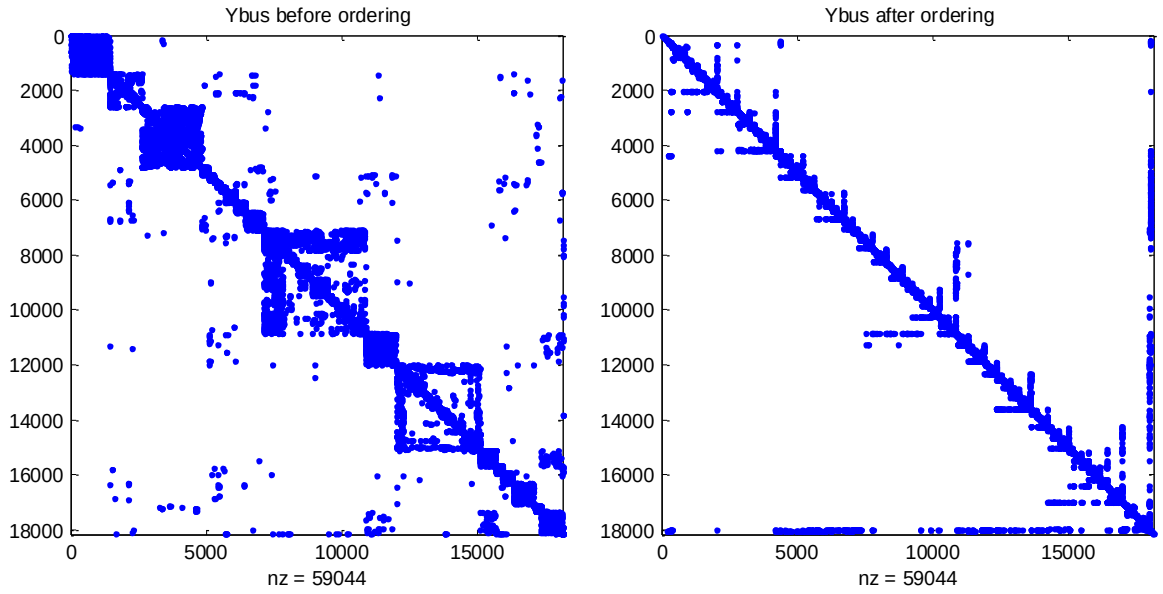


Figure 5: The left hand side figure shows the network matrix for the WECC interconnection; the right hand side figure shows the same matrix after ordering.

Both matrices are factorized in order to verify the effectiveness and impact of the ordering algorithm and the results are shown in Figure 6. The differences are visible to the naked eye and the reader can easily notice that the matrix on the right hand side has a smaller number of non-zero elements than the matrix on the left hand side. The actual number of non-zero elements is 1470223 on the left hand side matrix, versus 52126 in the matrix on the right.

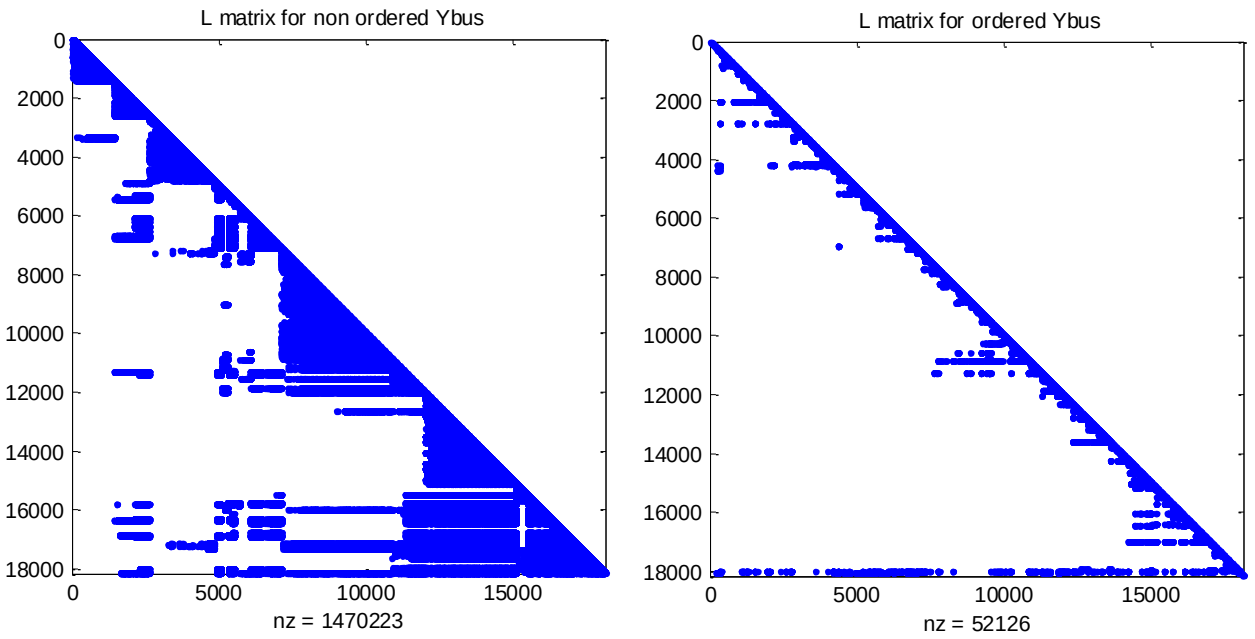


Figure 6: The left hand side matrix shows the L factor for the non-ordered WECC network matrix; the right hand side matrix shows the L factor for the ordered version of the WECC network matrix

Such large difference in the total number of non-zero elements, nearly 30 times, will significantly impact the calculation speed during the forward and backward solve due to the difference in the number of flops performed. This substantial difference will directly impact the calculation speed of the entire simulation. Therefore, choosing an appropriate ordering algorithm is a critical step in enhancing the simulation speed of power system simulations.

3.3.5 Solver comparison

Before replacing the current PSLF linear solver engine, a comparative evaluation process was performed using real size systems matrices obtained from modeling the WECC system. The solvers included in the testing stage were predefined after a screening procedure based on recent publications in the area of linear system solutions, which significantly narrowed down the most efficient solvers for circuit simulations. For confidentiality reasons, the name of the selected solver and ordering algorithms cannot be share at this moment.

A total of 4 ordering algorithms and 2 factorization algorithms were combined and tests, for a total combination of 8 pairs factorization-ordering algorithm. A comparative analysis indicated that the fastest factorization algorithm is nearly 50 times faster than the current PSLF factorization algorithm. On the other hand, the best ordering algorithm can reduce the amount of non-zeroes in one of the triangular factors by 16.4% in comparison to the PSLF ordering method.

Another important conclusion from exercise is that the current ordering algorithm used in PSLF is very efficient in terms of reducing the number of fill-ins. Despite the fact that it was written nearly 30 years ago, it outperformed recently written methods in terms of the total number of fills ins. On the other hand, the PSLF ordering algorithm takes more time to perform the ordering than recent

ordering methods. This difference can be partially explained by the fact that other tasks are performed by the ordering algorithm rather than simply ordering the network matrix.

The next step in the project will focus at implementing the selected solver and ordering algorithm in the PSLF core and ensuring that the program solution remains as accurate as when the native solver was used. This is an extreme challenging step given the fact that PSLF was not written with the required modularity to make this a trivial exercise and it has never been attempted before.

3.3.6 Selected solver implementation

In order to implement the selected solver as the main solution engine of PSLF, several intermediate steps needed to be executed. Firstly, the network admittance matrix needed to be passed to the solver along with the current injection from the network models. In addition to that, the Ybus matrix needed to be compressed into a compact sparse column (CSC) format.

Code to output the Ybus matrix had to be modified in order to be able to explicitly extract the network matrix from the core of PSLF algorithm. After extraction, the matrix needs to be compressed so that it is in a form that is suitable to be passed into the solver. Once the matrix is extracted and passed to the solver, a solution is obtained and the PSLF data structures get populated before the next integration time step. In order to enable this process, functionalities that were previously performed inside the PSLF solver had to be coded externally so that the solver replacement becomes a reality.

Preliminary results to verify solution accuracy after solver replacement are shown in Figure 7 and Figure 8 below. The figures show voltage magnitude at a particular bus in a 9 bus test case when subjected to a fault applied at 0.1s and cleared at 0.2s. Simulation is run for 10s and it can be concluded visual inspection that the solution accuracy of the new solver is the same, or very close, to the accuracy of the PSLF native solver.

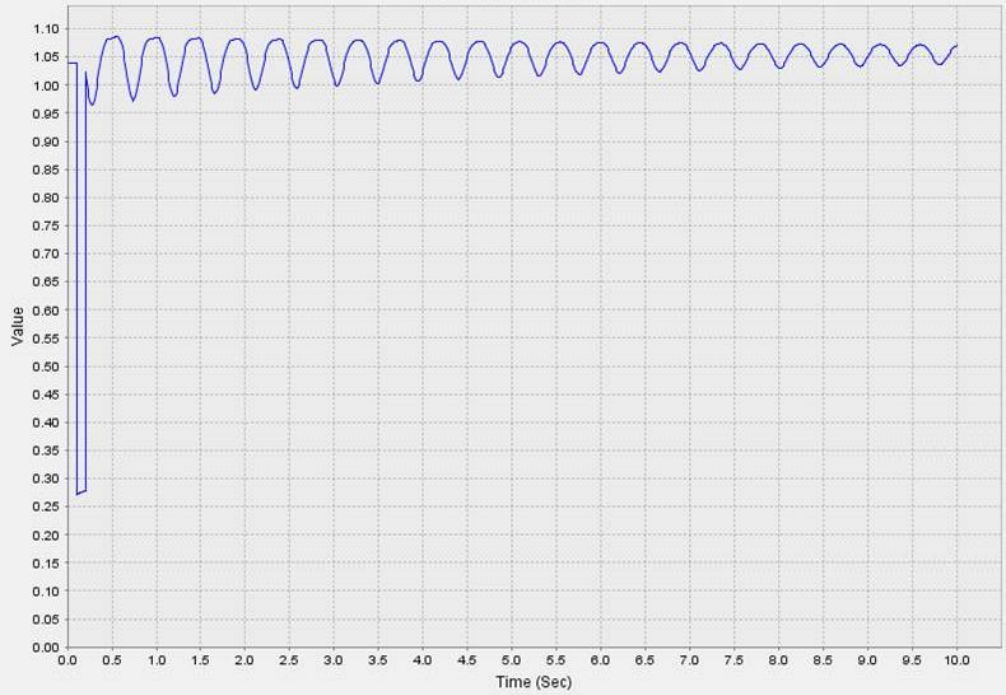


Figure 7: Dynamic response of a bus voltage for a 9 bus case with the new solver

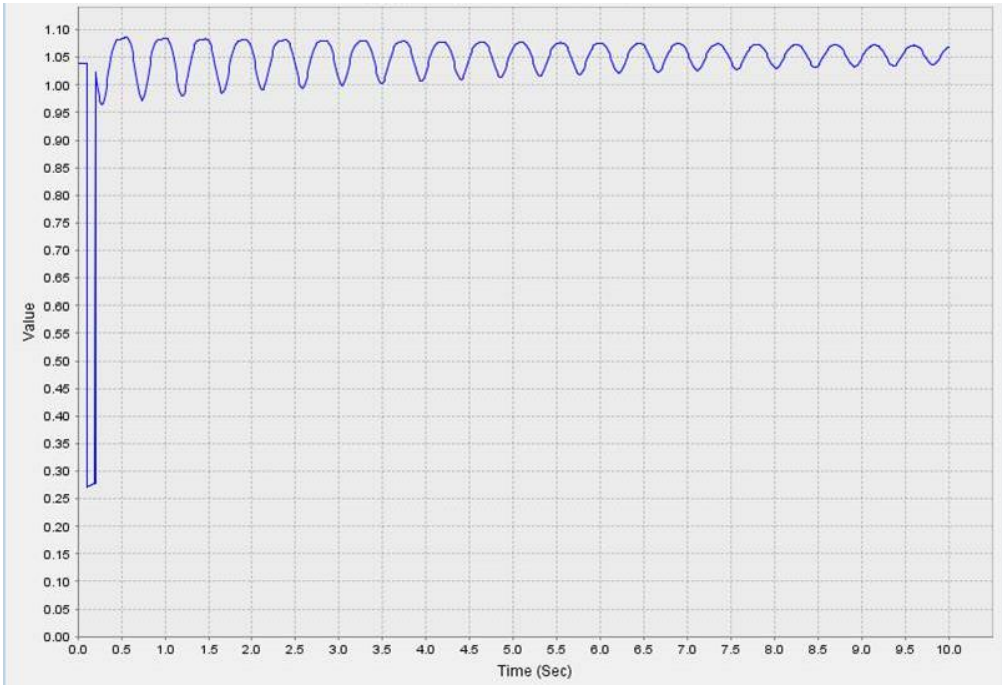


Figure 8: Dynamic response of a bus voltage for the 9 bus test case with the PSLF native linear solver

Additional tests to verify the solver accuracy have been performed in the WECC test case with 18202 buses. Figure 9 and Figure 10 below show the solution differences for the voltage magnitude and angle between the newly selected linear solver and the PSLF native solver.

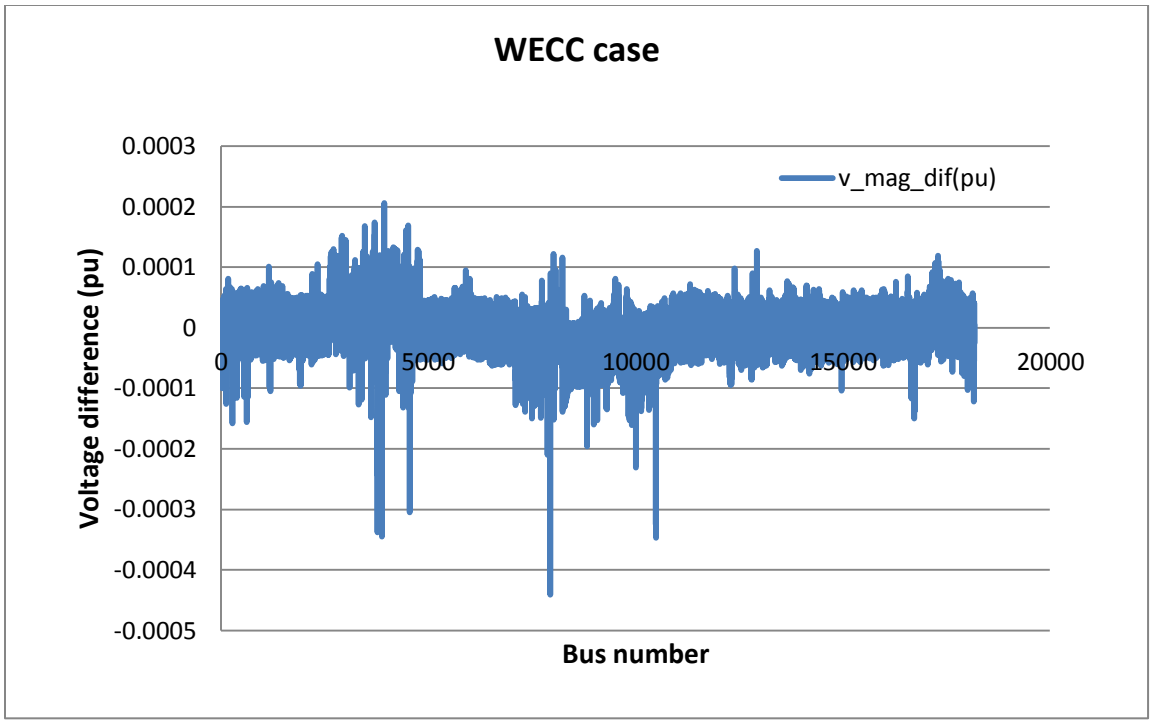


Figure 9: Voltage magnitude differences between the PSLF solver and the new solver showing that the solution differences are minimal.

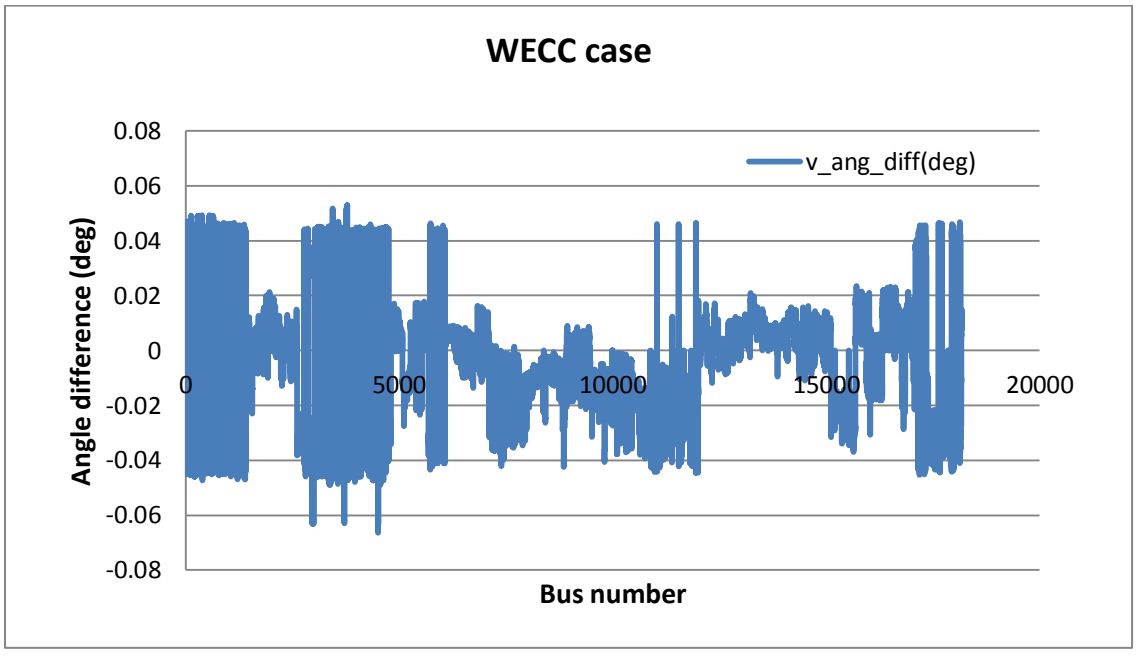


Figure 10: Voltage angle differences between the PSLF solver and the new solver showing that the solution differences are minimal.

It can be observed that the solution differences are minimal and will not significantly impact the results, as the plots shown in Figure 7 and Figure 8 had already indicated. In fact, the voltage magnitude solution is the identical in both solvers up to the 3rd significant digit, with most buses

showing a difference les or equal to of 0.0001pu. The angle differences are identical up to the 1st decimal place, with most differences being within ± 0.04 degrees. Although present, these differences are considered to be negligible because they do not impact the dynamic response of the system or its performance to a significant extent.

3.4 Simulation results with new linear solver and parallel solution engine

A comparison between the existing PSLF version and the version with a new linear solver and parallel solution engine is presented in this section. Results were obtained on a 4 core/8threads laptop computer and a PNNL supercomputer with 8 cores and 20 threads per core.

3.4.1 Test case

In order to verify speed gains on larger systems, it is critical to ensure that the test case being simulated is large enough (has enough flops) so that the impact of a parallel implementation is readily visible. This is of primary importance because of the overhead costs involved with forking-joining threads at the end of each parallel block may overcome any gains.

Therefore, an Eastern Interconnect (EI) case was chosen as the base case and will be used as benchmark for the time domain simulations performed herein. This database is an excellent choice for many reasons, the main ones being:

1. It is real database representing the eastern interconnect of the US power grid and is used daily by several transmission planners and system operators. A successful demonstration using this database indicates that a commercial version of the parallel prototype is very close to reality.
2. It is the largest available case and should be able to harvest the highest benefits from a parallel version of PSLF
3. The actual validation of the parallel version of PSLF on an actual database using traditional increases the credibility of the results

Table 4 below contains a list of all models present in the test case utilized, including a count of each individual model. A total of 20540 dynamic models are present in this database, which has nearly 70k buses. More details of the system can be found in in Table 2.

Table 4: Model count and names of all models initialized and simulated on the EI database

Model name	Model Count	Model name	Model Count
alwscc	34	**genind**	214
crcmgv	17	**genrou**	5106
esac1a	275	**gensal**	1604
esac2a	62	**gewtg**	111
esac3a	26	**ggov1**	741
esac4a	101	**gpwscc**	61
esac5a	264	**hyg3**	79
esac6a	36	**hygov**	622
esac7b	234	**hygov4**	18
esac8b	279	**ieeeg1**	1399
esdc1a	1353	**ieeeg3**	31
esdc2a	181	**ieeest**	173
esdc3a	36	**lsdt1**	126
esdc4b	7	**pidgov**	55
esst1a	941	**pss2a**	1055
esst2a	52	**pss2b**	71
esst3a	24	**rexs**	50
esst4b	808	**scrx**	318
esst5b	6	**sexs**	191
esst6b	38	**stcon**	13
exac1a	9	**tgov1**	690
exac2	233	**tgov3**	14
exac3	31	**vwscc**	51
exac3a	3	**wlwscc**	1
exbbc	3	**wndtge**	109
exdc2	2	**wsccst**	31
exdc4	123	**wt1g**	7
exeli2	8	**wt1p**	7
exivo	2	**wt1t**	7
expic1	168	**wt2e**	6
exst2	171	**wt2g**	6
exst3	54	**wt2p**	6
exwtge	111	**wt2t**	6
g2wscc	9	**wt4e**	12
gast	1167	**wt4g**	12
gencls	657	**wt4t**	12
		Grand Total	**20540**

3.4.1.1 Results on laptop computer

Figure 11 and Figure 12 show the reduction in runtime and the speedup factor for various numbers of threads. These results were obtained on a laptop computer with 4 cores and 8 threads available. It can be observed that the runtime can be reduced by a factor slightly greater than 2, while saturation in performance seems to occur at around 4 threads.

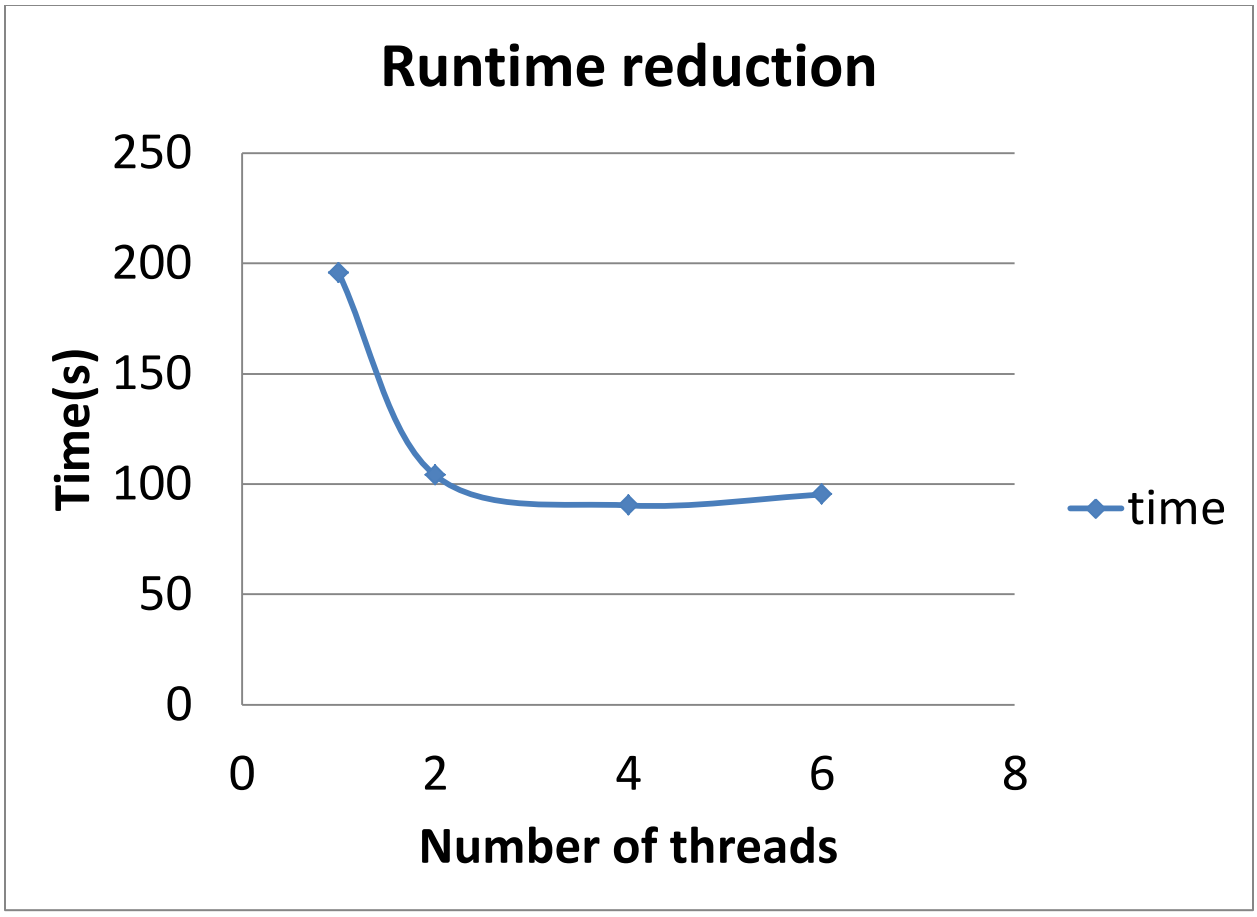


Figure 11: Simulation gains for 20 s simulation with fault applied at 0.2s and cleared at 0.3s

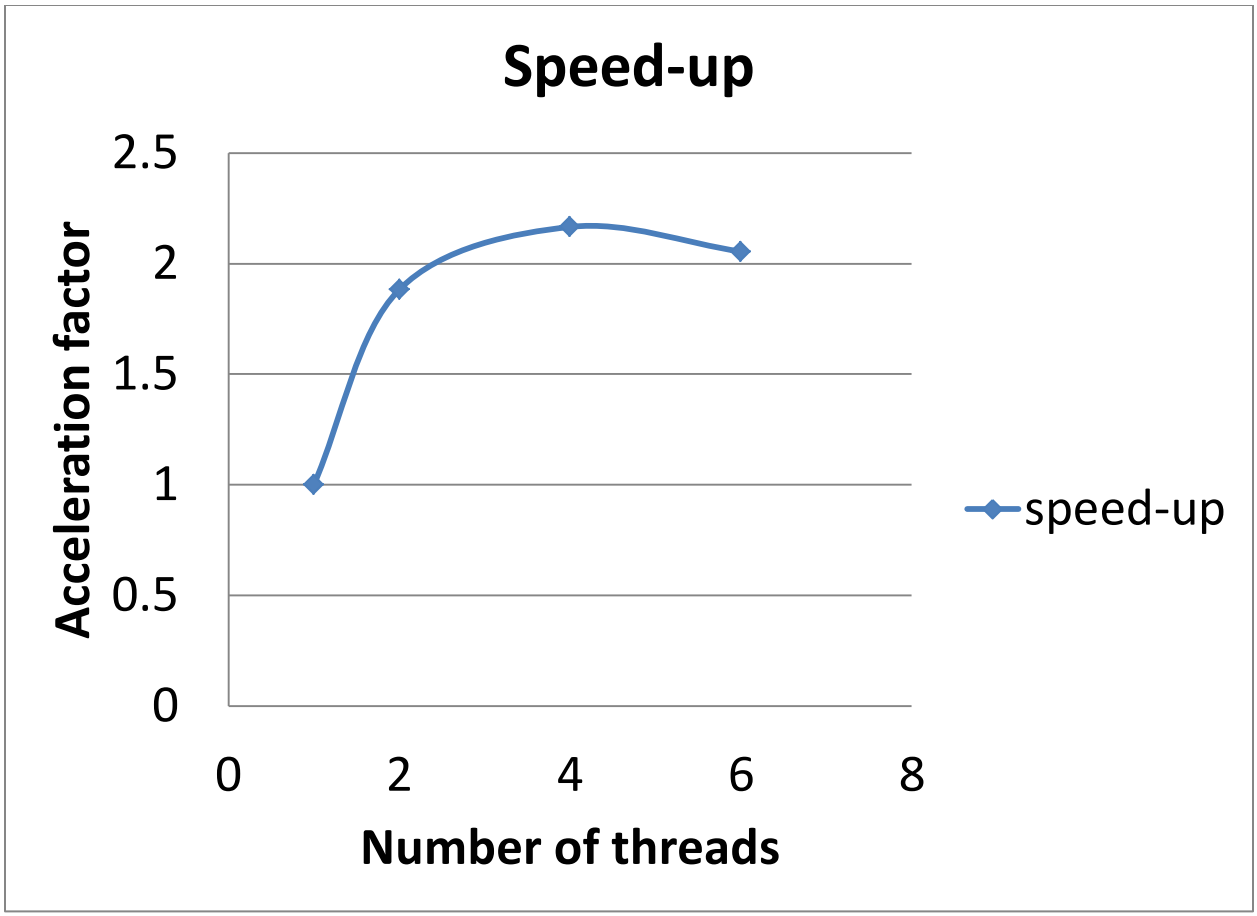


Figure 12: Acceleration factor for 20 s simulation with fault applied at 0.2s and cleared at 0.3s showing a speedup of nearly 2.17x.

In order to ensure that the accuracy of the results remains unchanged, several variables at the faulted generator are plotted for various configurations. The dynamic response of the single thread engine with native PSLF solver is identical to the response of the 2, 4 and 6 threads version of PSLF with the new linear solver. The simulation results for all different versions of PSLF are shown from Figure 50 to Figure 57 in Appendix C.

3.4.1.2 Results on PNNL PIC computer

The same tests performed in the laptop computer were successfully reproduced in the PNNL PIC machine in order to verify their accuracy.

Figure 13 and Figure 14 show that the runtime simulation of a typical 20s simulation on PNNL's PIC computer can also be reduced by slightly over half, with a highest speedup gain of **2.22x** at 12 cores.

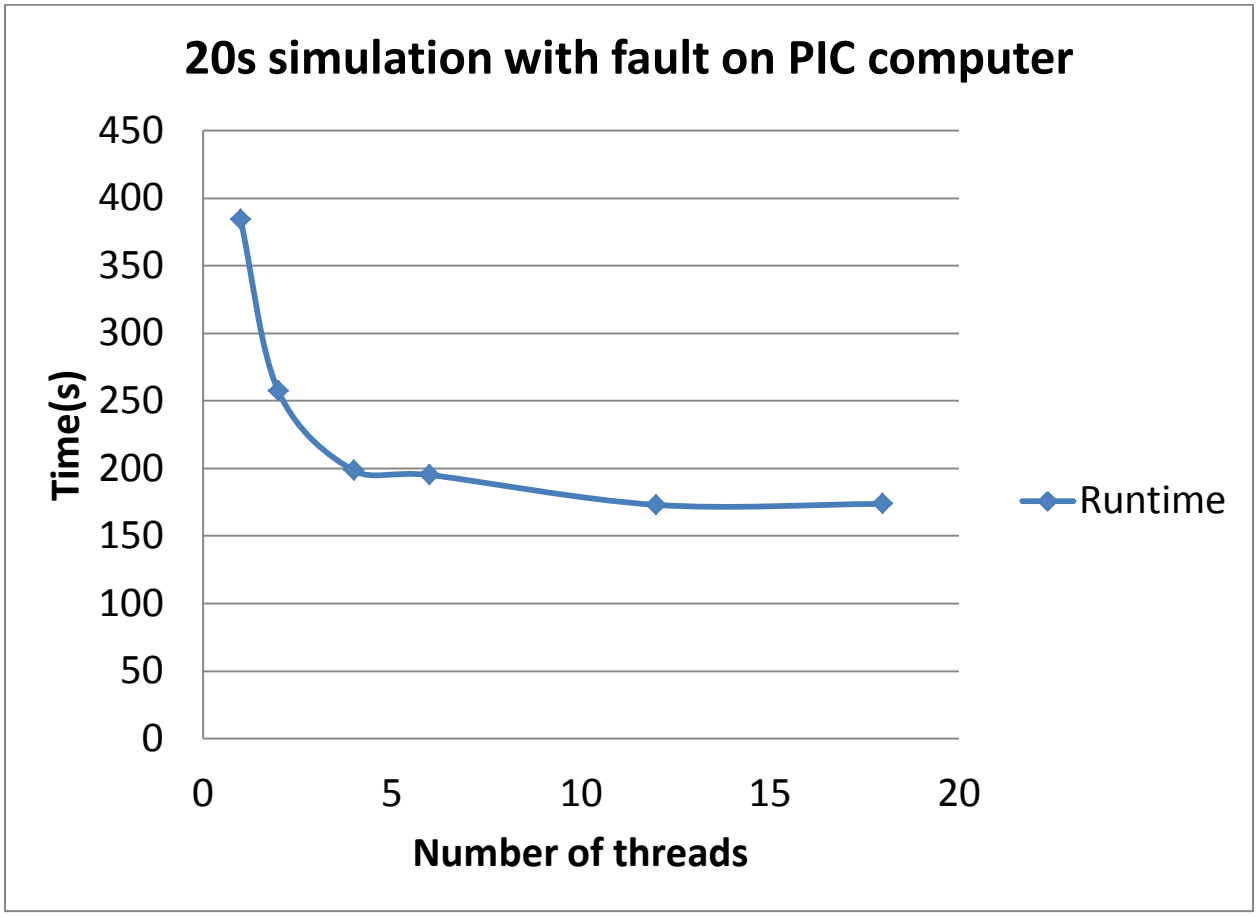


Figure 13: Simulation runtime reduction for typical 20 s simulation with fault applied at 0.2s and cleared at 0.3s

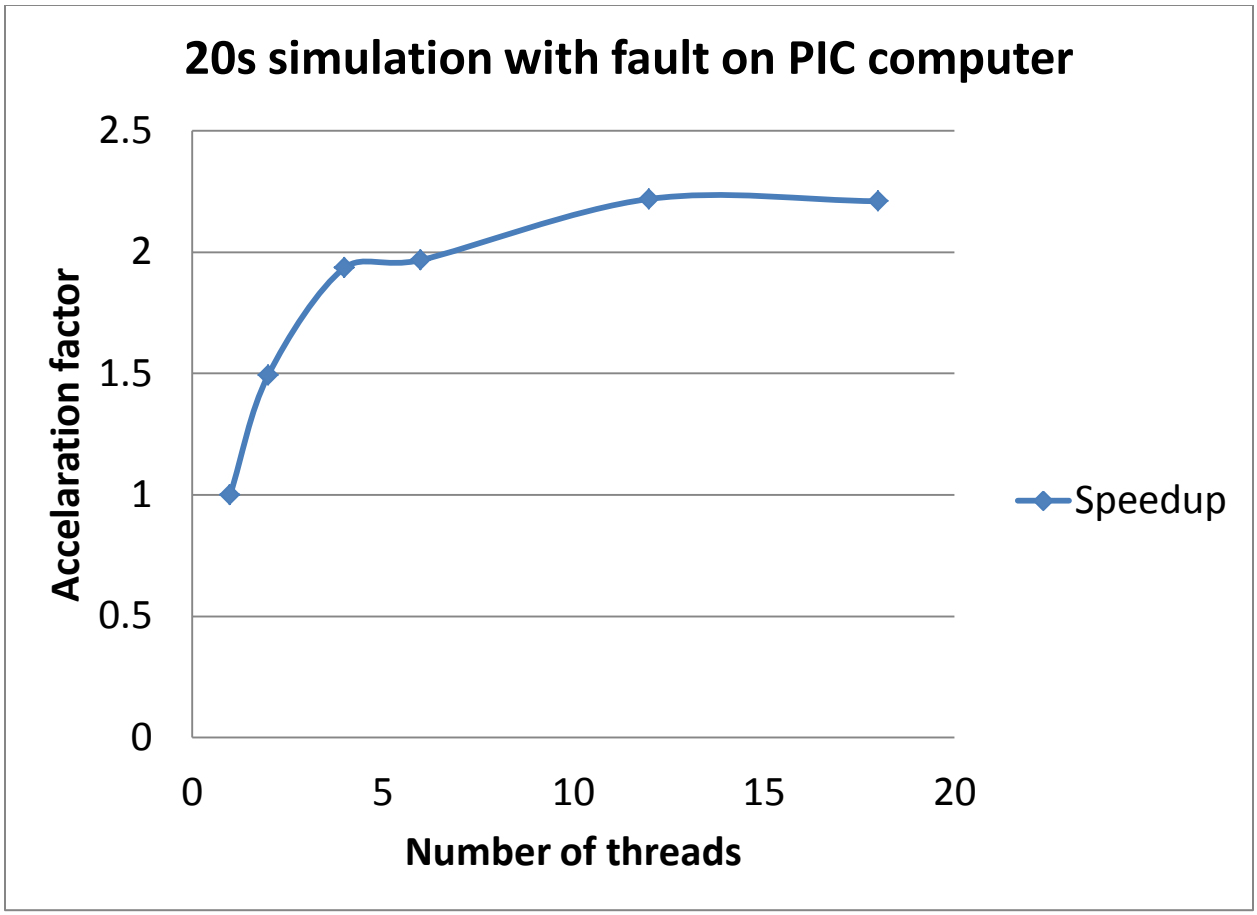


Figure 14: Acceleration factor for 20 s simulation with fault applied at 0.2s and cleared at 0.3s showing a speedup of nearly 2.22x.

In this case, the computer had 8 processors with 20 threads each. A total of 6 simulations considering 1, 2, 4, 6, 12 and 18 threads were performed. The variables that vary the most during the simulation are plotted for verification. Results are shown from Figure 58 to Figure 69 in Appendix D and indicate that the accuracy does not change with the number of threads used and the presence of the new linear system solver.

3.5 Conclusions

In this work, a full overhaul of the dynamic simulation engine of GE PSLF was performed. The modifications include the parallel solution of differential equations, new and improved ordering, factorization and forward/backward substitution algorithms. Speed improvements have been observed for both typical laptop computers as well as a PNNL supercomputer. Speed gains are observed in all sections of the program responsible for dynamic simulation, with the speed gains varying from section to section. Speed gains on each section are not reported in detail due to proprietary reasons.

Another important observation is that although a significant speed improvement was obtained, it is unlikely that major speed gains will occur under the existing program architecture. This is mainly due to the fact that all code used in the dynamic simulation core was reviewed and improved, leaving

very little room for further improvements. Moreover, despite the fact that the largest power system model in the US has nearly 100 thousand buses, the problem size that arises from its formulation is still relatively small for parallel iterative linear solver to be used. As the problems dimension grows larger and the number of dynamic models increases, it is expected that iterative solvers will outperform direct methods and the performance of the program under the existing architecture may benefit from that. Given the fact that most commercial transient stability simulators have a similar architecture as PSLF, it is unlikely that very high speed gains are obtained on their architectures as well. For additional speed improvements due to parallelization, more significant changes in the solution architecture and perhaps integration time step will be required.

Overall, a typical 20s simulation with a fault applied and cleared can be sped up over **2x** on a computer with at least 4 cores and 8 threads. These results are promising and the possibility that they can become part of GE PSLF simulator in a short term is substantial.

4 Simultaneous time stacking method for fast dynamic simulation

As discussed in Section 2, the significant improvement in the solution speed of the power system dynamic simulation need alternate DAE solution architecture that are multi-core friendly. Development of such architecture requires significant research and development. To overcome the challenge, PNNL research team worked closely with GE GRC and Energy Consulting teams to develop faster-than-real-time dynamic simulation to enable predictive capabilities for validating small signal stability controls by exploring a variety of mathematical algorithms and advanced computational methods using high performance computing (HPC) techniques. The main idea is to develop methods that have the best fit of the algorithms and computational libraries for the power grid simulation problem. PNNL's main tasks include:

(1) Support the development of parallel version of PSLF
    a. Identification of fast linear solvers that can be used by the existing PSLF software
    b. Providing test systems, data, HPC resources, algorithms and benchmarking as needed
(2) Develop a new "time-stacking" method to obtain system trajectories within a time window simultaneously
    a. Formulation of the implicit integration
    b. Formulation of the "time-stacking" method
    c. Identification of fast solvers to gain speedup
    d. Investigation of adaptive time steps in dynamic simulation

With faster dynamic simulation, a variety of benefits are expected, as it will:

(1) Enable online security assessment to enhance situational awareness by Estimating the current and near-future system status more accurately and comprehensively
(2) Expedites system planning process by significantly reducing simulation time from weeks to hours, or even minutes
(3) Prevents cascading failures by validating preventive and corrective control schemes ahead of time
(4) Enables real-time path rating by improving transfer capability of existing lines, relieving transmission congestion, facilitating emergency control to integrate more renewable energy and deferring the construction of new transmission lines
(5) Further improves power grid reliability

4.1 Formulation of dynamic simulation

The dynamics of a power system can be represented by a set of first-order differential and algebraic equations:

$$\dot{x} = f(x, y) \quad (2)$$

$$0 = g(x, y) \quad (3)$$

where the boldface denotes vector, x, represents a vector of state variables (e.g., rotor angle and speed of a generator), and y represents a vector of algebraic variables (e.g., bus voltage magnitudes

and angles). A power system described in (2) and (3) can be solved using various solution schemes [20]. The main differences between the schemes rest in the way the differential and algebraic equations are interfaced (partitioned or simultaneous), and in the numerical integration method approach (implicit or explicit). The partitioned-solution method is predominantly used in industrial programs, which are mainly designed for sequential computation on a single core computer. In this method, differential equation set (2) is algebrized and then solved for state variables. The algebraic equation set (3) is then solved for algebraic variables. These solutions are alternated with each other in some manner inside each time step. In the simultaneous solution method, (2) is algebrized and then lumped together with (3) to form a larger set of algebraic equations to be solved. This method mostly adopts implicit integration methods particularly the trapezoidal rule, which seems to be a most reliable and elegant method for the numerical solution of practical systems of differential equations [20], [21].

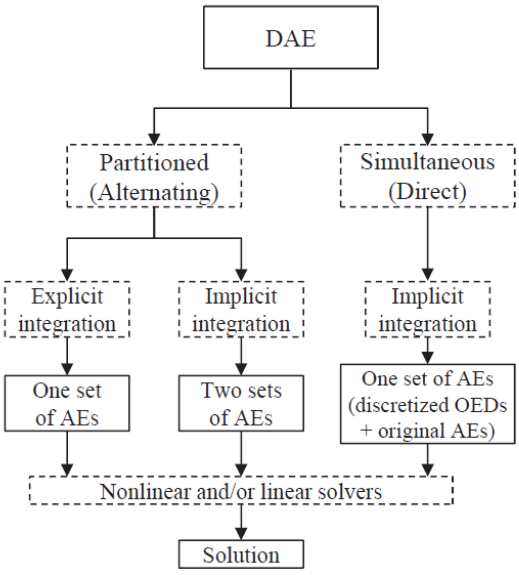


Figure 15. Categories of solution approaches for dynamic simulation

4.1.1 Partitioned method with explicit integration

Using an explicit integration method, (1a) is converted into an algebraic equation set

$$x_Y = F(x_{Y-1}, y_{Y-1}) \quad (4)$$

where Y is the current simulation step. Equation (4) can be solved for $\boldsymbol{x}_Y$, which is then used to solve (5) for y_Y.

$$0 = g(x_Y, y_Y) \quad (5)$$

The explicit methods evaluate the state variables of the current step explicitly as a function of the values at previous steps. Equations (2) and (3) are solved alternatively at each time step. As mentioned above, a small time step is required in the explicit methods to guarantee numerical stability so that the total simulation time is relatively longer.

4.1.2 Partitioned method with implicit integration

Using an implicit integration method, equation (2) is converted into an algebraic equation set

$$x_Y = F(x_Y, y_Y) \quad (6)$$

The initial guess of x_Y can be obtained based on x_{Y-1} and y_{Y-1} using any explicit integration formula. The value of x_Y is used to solve equation (5) for y_Y. The value of y_Y is then used in equation (6) to obtain x_Y, which is used in equation (5) to solve y_Y. This process will be repeated until the solution is converged.

4.1.3 Simultaneous method with implicit integration

Writing equation (6) in the form

$$0 = H(x_Y, y_Y) \quad (7)$$

and then combining equation (7) and equation (5), one can obtain a single set of algebraic equations. The Newton's method can be used to solve this algebraic equation set, requiring the construction and solution of the Jacobian matrix at each iteration:

$$\boldsymbol{J} = \begin{bmatrix} \boldsymbol{J}_{Fx} & \boldsymbol{J}_{Fy} \\ \boldsymbol{J}_{gx} & \boldsymbol{J}_{gy} \end{bmatrix} \quad (8)$$

When ignoring the sub-matrices $\mathbf{J}_{Fy}$ and $\mathbf{J}_{gx}$, and updating **F** and **g** alternately, this method degenerates to partition method with implicit integration. No matter which scheme is used, those non-linear algebraic equations need to be solved, using either Gauss or Newton (more general) type procedures. Dishonest Newton or very dishonest Newton methods are often used in industrial softwares to increase computation speed. When Newton-like method is used, linear solvers are required. A summary of preconditioning techniques for large linear systems can be found in [22]. As an example, Khaitan and McCalley applied the multi-frontal method in linear solvers [23].

4.2 A time-stacking method

In the proposed research work, PNNL team first implemented the "simultaneous method" using HPC platform to simultaneously solve one set of algebra equations at each time step, which contain both the discretized differential equations and the network algebraic equations. The formulation of the problem is formulated in Equation (9). Since implicit integration method has better numerical stability, a larger time step can be used to save the total computation time. The flowchart for the "simultaneous method" is shown in Figure 16.

$$\begin{cases} x_{\gamma+1} = f(x_\gamma, x_{\gamma+1}, y_\gamma, y_{\gamma+1}) \\ 0 = g(x_{\gamma+1}, y_{\gamma+1}) \end{cases} \quad (9)$$

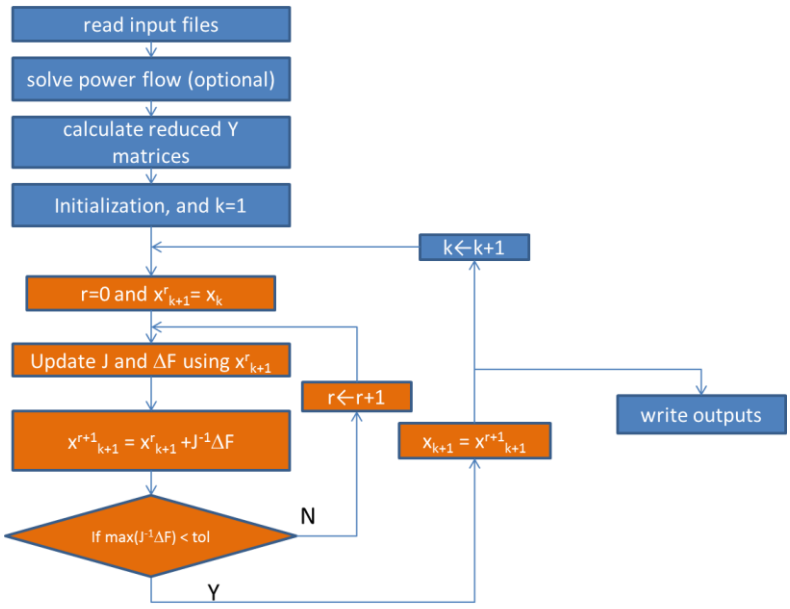


Figure 16. Flowchart of the "simultaneous method"

To further increase computational speed of dynamic simulation, a "time-stacking" method is proposed and implemented to compute system dynamic behavior several time steps within a time frame simultaneously using HPC techniques, instead of computing state variables at each time step in Equation (9). The key idea is shown in the Figure 18. Although the size of the problem is increased due to the fact that solving state variables for several time steps are combined together into one set of algebraic equations, the overall solution time can be greatly reduced with the help of advanced parallel linear solvers that are commonly used in the HPC platform.

$$\begin{cases} x_{k+1} = f(x_k, x_{k+1}, y_k, y_{k+1}) \\ 0 = g(x_{k+1}, y_{k+1}) \end{cases}$$

$$\downarrow$$

$$\begin{cases} x_{k+2} = f(x_{k+1}, x_{k+2}, y_{k+1}, y_{k+2}) \\ 0 = g(x_{k+2}, y_{k+2}) \end{cases}$$

$$\downarrow$$

$$\dots$$

$$\begin{cases} x_{k+m} = f(x_{k+m-1}, x_{k+m}, y_{k+m-1}, y_{k+m}) \\ 0 = g(x_{k+m}, y_{k+m}) \end{cases}$$

$$\begin{cases} x_{k+1} = f(x_k, x_{k+1}, y_k, y_{k+1}) \\ 0 = g(x_{k+1}, y_{k+1}) \\ x_{k+2} = f(x_{k+1}, x_{k+2}, y_{k+1}, y_{k+2}) \\ 0 = g(x_{k+2}, y_{k+2}) \\ \dots \\ x_{k+m} = f(x_{k+m-1}, x_{k+m}, y_{k+m-1}, y_{k+m}) \\ 0 = g(x_{k+m}, y_{k+m}) \end{cases}$$

Figure 17: Sequential time-stepping process (left) vs. a new time-stacking method (right).

4.3 Identification of efficient linear solvers

To solve the time-stacking problem (or more generally, *Ax=b*) accurately and rapidly, find an ideal solver is the key. Iterative methods are required due to the implicit nature of the problem, which are to develop an iterative procedure starting from an initial approximation to converge to the solution, e.g., the Newton's method. The two main classes of iterative methods are the stationary iterative methods, and the more general Krylov subspace methods. The Krylov subspace methods work by forming a basis of the sequence of successive matrix powers times the initial residual (the Krylov sequence). The approximations to the solution are then formed by minimizing the residual over the subspace formed. The prototypical method in this class is the conjugate gradient (CG) method. Iterative methods like CG are more suitable for use with sparse matrices in power system dynamic simulation than direct methods. Most iterative methods are memory-efficient and run quickly with sparse matrices. In the last few decades the CG method has been used successfully in various engineering applications for the solution of linear algebraic equations. The convergence rate of CG iterative solution depends on the condition number. When the matrix's condition number is minimized, the method usually converges much faster. Preconditioning is a technique for minimizing the condition number by pre-multiplying both sides of equation by the inverse of a preconditioner matrix, *p*,

$$p^{-1}Ax = p^{-1}b \quad (10)$$

to yield a new equation of

$$\hat{A}x = \hat{b} \quad (11)$$

where the condition number of $\widehat{\mathrm{A}}$ is much smaller than the original matrix of A. This method is known as Preconditioned Conjugated Gradient (PCG) method. PCG method presents great potential for achieving high performance in parallel computers. Parallelization can be implemented at both the preconditioner generation step and the equation solving step. Existing software libraries for iterative method include Hypre, PETSc, and Aztec, etc. We have applied the PCG method to solve power system state estimation problem utilizing the Hypre software package. With the help of ILU preconditioner and parallel computers, the execution time of solving the linear algebraic equation can be greatly reduced using multiple processors. It's promising to see that by applying the parallelized PCG method to solve the network algebraic equation, the solution time for power system dynamic simulation can also be significantly reduced to achieve a faster than real time dynamic simulation. Other similar types of linear solvers will be investigated and compared to achieve the best possible performance.

4.4 Adaptive time stepping method

An adaptive time stepping method is also proposed in order to further speed up dynamic simulation by using larger time steps when the power system is at quasi-steady state and using smaller time step at/near switching conditions. As an example, a smaller time step is used to simulate the detailed dynamic behavior of the system during a fault condition; and a larger time step can be used at initial conditions or after a disturbance is settled. In this implementation, the time step of dynamic simulation is adjusted based on local truncation errors, the performance of Newton corrector

iteration, and switching events and faults. The time step is increased only if the max norm of the local error vector is below a specified tolerance for a number of time steps. On the other hand, if the tolerance is exceeded, then the time step is reduced and fixed to the reduced value for a number of time steps before attempting to increase it.

4.5 Methodology Implementation and Observed Performance

4.5.1 The time-stacking method with classical generator model

When generators are represented using the classical model, the equations of motion for generator *i* in per unit are:

$$\begin{cases} \dot{\delta}_{(i)} = \omega_b \, \omega_{(i)} \\ \dot{\omega}_{(i)} = \dfrac{1}{2H_{(i)}} \left(P_{m(i)} - P_{e(i)} - D_{(i)} \omega_{(i)} \right) \end{cases} \tag{12}$$

where $\omega(i)$ is the per unit speed deviation for generator *I*; $\delta(i)$ is the angular position of the rotor of generator *i* in electrical radians with respect to a synchronously rotating reference; $H(i)$ is the inertia constant of generator *i* using system base; $D(i)$ is the damping factor or coefficient of generator *i* in pu torque/pu speed deviation; $P_m(i)$ is the mechanical power input of generator *i*, and ω_b is the base rotor electrical speed in radians per second. In this case, state variables in (2) only include rotor speed and angle, i.e.,

$$\mathbf{x} = \left[\delta_{(1)}, \omega_{(1)}, \cdots, \delta_{(i)}, \omega_{(i)}, \cdots, \delta_{(n)}, \omega_{(n)} \right]^{\mathrm{T}} \tag{13}$$

where *n* is the number of generators. Using the classical model to study the transient stability for a multi-machine system, it is often assumed [24]:

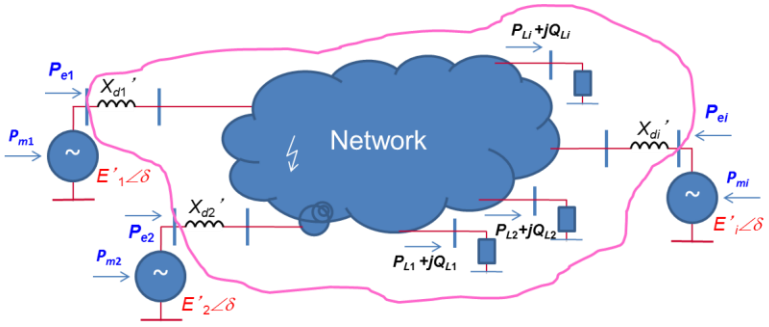


Figure 18. Illustration of a power network represented by classical generator model and constant impedance load

1) Mechanical power input, $P_m(i)$ is constant.

2) The mechanical rotor angle of a machine coincides with the angel of the voltage behind the transient reactance.

3) The network is assumed to be in the sinusoidal steady state.

4) Load is represented by the passive impedances, i.e., the dynamics of load is ignored.

Therefore, let *Y''* (with a dimension equal to *m* by *m*) denote the nodal admittance matrix of an *m*-bus system comprised of *n* generator buses and *m-n* load buses. One can follow the steps in [24] or [25] to add machine internal buses and include load impedance into the network admittance matrix, resulting in an extended Y-bus matrix — *Y'* (with a dimension of *m+n* by *m+n*). The network equation becomes:

$$\begin{bmatrix} \mathbf{I}_E \\ \mathbf{0} \end{bmatrix} = \mathbf{Y}' \begin{bmatrix} \mathbf{E} \\ \mathbf{V} \end{bmatrix} \quad (14)$$

Where $\mathrm{Y}' = \begin{bmatrix} \mathrm{Y_{nn}} & \mathrm{Y_{nm}} \\ \mathrm{Y_{mn}} & \mathrm{Y_{mm}} \end{bmatrix}$.

The network equations can be reduced to

$$\mathrm{IE} = \mathrm{Y} * \mathrm{E} \quad (15)$$

where $Y = Ynn - Ynm\ Ymm^{-1}\ Ymn$. Numerical evaluation of direct solvers for large sparse, symmetric linear equations can be found in the literature, which could be used to solve $Ymm^{-1}Ymn$. For each generator internal voltage bus i, the injection current in system reference can be expressed as :

$$\overline{I}_{(i)}^{sys} \triangleq I_{\mathrm{re}(i)} + jI_{\mathrm{im}(i)} = \sum_{k=1}^{n} \overline{Y}_{(ik)} \overline{E}_{(k)} \quad (16)$$

where a bar above the notation represents a complex number, $\bar{I}_i^{sys}$ is the injection current of bus *i* in system reference, and $\bar{E}(k)$ is the stator internal voltage of bus k in system reference. Let $\bar{E}(k) = E_{re}(k) + jE_{im}(k)$ and $\bar{Y}_{(ik)} = G_{(ik)} + jB_{(ik)}$, resolving (16) into real and imaginary part yields,

$$I_{\mathrm{re}(i)} = \sum_{k=1}^{n} \left[E_{\mathrm{re}(k)} G_{(ik)} - E_{\mathrm{im}(k)} B_{(ik)} \right] \quad (17)$$

$$I_{\mathrm{im}(i)} = \sum_{k=1}^{n} \left[E_{\mathrm{re}(k)} B_{(ik)} + E_{\mathrm{im}(k)} G_{(ik)} \right]$$

The transformation from machine dq0 reference frame to system reference frame is

$$\begin{bmatrix} \mathrm{re} \\ \mathrm{im} \end{bmatrix} = \begin{bmatrix} \sin\delta & \cos\delta \\ -\cos\delta & \sin\delta \end{bmatrix} \begin{bmatrix} d \\ q \end{bmatrix} \quad (18)$$

In the classic generator model, since $E'_d = 0$, applying the above transformation to I_{re}, I_{im}, E_{re}, *and* E_{im} in (17) and after manipulations, we obtain equations in machine reference frame

$$
\begin{aligned}
0 &= \sin\delta_{(i)} I_{d(i)} + \cos\delta_{(i)} I_{q(i)} - \sum_{k=1}^{n} \left[\left(\cos\delta_{(k)} E'_{q(k)} \right) G_{(ik)} - \left(\sin\delta_{(k)} E'_{q(k)} \right) B_{(ik)} \right], \\
0 &= -\cos\delta_{(i)} I_{d(i)} + \sin\delta_{(i)} I_{q(i)} - \sum_{k=1}^{n} \left[\left(\cos\delta_{(k)} E'_{q(k)} \right) B_{(ik)} + \left(\sin\delta_{(k)} E'_{q(k)} \right) G_{(ik)} \right]
\end{aligned}
\tag{19}
$$

The active power at air gap can be expressed as

$$
P_{e(i)} = E'_{q(i)} I_{q(i)} \tag{20}
$$

Replacing $P_e(i)$ in (12) by (20) yields

$$
\begin{cases}
\dot{\delta}_{(i)} = \omega_b \, \omega_{(i)} \\
\dot{\omega}_{(i)} = \dfrac{1}{2H_{(i)}} \left(P_{m(i)} - E'_{q(i)} I_{q(i)} - D_{(i)} \omega_{(i)} \right)
\end{cases}
\tag{21}
$$

Combining (21) and (19) results in a set of DAE in the same form as (2) and (3), where

$$
\mathbf{x} = \left[\delta_{(1)}, \omega_{(1)}, \cdots, \delta_{(i)}, \omega_{(i)}, \cdots, \delta_{(n)}, \omega_{(n)} \right]^{\mathrm{T}} \tag{22}
$$

$$
\mathbf{y} = \left[I_{d(1)}, I_{q(1)}, \cdots, I_{d(i)}, I_{q(i)}, \cdots, I_{d(n)}, I_{q(n)} \right]^{\mathrm{T}} \tag{23}
$$

and the other notations represent parameters.

We first implemented the trapezoidal method to run dynamic simulations with classical generator model. For time step Υ, applying trapezoidal rule to (17), and then combining the resulted algebraic equations with (19), yields

$$
\begin{cases}
0 = \delta_{(i,\gamma)} - \delta_{(i,\gamma-1)} - \dfrac{h}{2} \left(\omega_b \, \omega_{(i,\gamma-1)} + \omega_b \, \omega_{(i,\gamma)} \right) \\
0 = \omega_{(i,\gamma)} - \omega_{(i,\gamma-1)} - \dfrac{h}{2} \left[\dfrac{1}{2H_{(i)}} \left(P_{m(i,\gamma-1)} - E'_{q(i)} I_{q(i,\gamma-1)} - D_{(i)} \omega_{(i,\gamma-1)} \right) \right. \\
\qquad \left. + \dfrac{1}{2H_{(i)}} \left(P_{m(i,\gamma)} - E'_{q(i)} I_{q(i,\gamma)} - D_{(i)} \omega_{(i,\gamma)} \right) \right] \\
0 = \sin\delta_{(i,\gamma)} I_{d(i,\gamma)} + \cos\delta_{(i,\gamma)} I_{q(i,\gamma)} - \sum_{k=1}^{n} \left[\left(\cos\delta_{(k,\gamma)} E'_{q(k)} \right) G_{(ik)} - \left(\sin\delta_{(k,\gamma)} E'_{q(k)} \right) B_{(ik)} \right] \\
0 = -\cos\delta_{(i,\gamma)} I_{d(i,\gamma)} + \sin\delta_{(i,\gamma)} I_{q(i,\gamma)} - \sum_{k=1}^{n} \left[\left(\cos\delta_{(k,\gamma)} E'_{q(k)} \right) B_{(ik)} + \left(\sin\delta_{(k,\gamma)} E'_{q(k)} \right) G_{(ik)} \right]
\end{cases}
\tag{24}
$$

where h is the integration step length. The first two equations correspond to equations of **F** in (4), and the last two correspond to equations of **g** in (5). Combining these equations results in

$$\mathbf{0} = \mathbf{H}(\mathbf{x}_\gamma, \mathbf{y}_\gamma) = \begin{bmatrix} F_1(\mathbf{x}_\gamma, \mathbf{y}_\gamma) \\ F_2(\mathbf{x}_\gamma, \mathbf{y}_\gamma) \\ \cdots \\ g_1(\mathbf{x}_\gamma, \mathbf{y}_\gamma) \\ g_2(\mathbf{x}_\gamma, \mathbf{y}_\gamma) \\ \cdots \end{bmatrix} \quad (25)$$

Where

$$\mathbf{x}_\gamma = [\delta_{(1,\gamma)}, \omega_{(1,\gamma)}, \cdots, \delta_{(i,\gamma)}, \omega_{(i,\gamma)}, \cdots, \delta_{(n,\gamma)}, \omega_{(n,\gamma)}]^{\mathrm{T}}$$

and

$$\mathbf{y}_\gamma = [I_{d(1,\gamma)}, I_{q(1,\gamma)}, \cdots, I_{d(i,\gamma)}, I_{q(i,\gamma)}, \cdots, I_{d(n,\gamma)}, I_{q(n,\gamma)}]^{\mathrm{T}}$$

The Newton's method can be used to solve the above equations. The corresponding Jacobian matrices are shown below:

$$\mathbf{J} = \begin{bmatrix} \mathbf{J}_{Fx} & \mathbf{J}_{Fy} \\ \mathbf{J}_{gx} & \mathbf{J}_{gy} \end{bmatrix} \quad (26)$$

Each sub-matrix is calculated below:

$$J_{Fx} = \mathrm{diag}(\mathbf{J}_{\mathbf{Fx}}^{\mathbf{i}})$$

$$\mathbf{J}_{Fx}^{i} = \begin{bmatrix} 1 & -h\omega_b/2 \\ 0 & 1 + hD_{(i)}/(4H_{(i)}) \end{bmatrix}$$

$$J_{Fy} = \mathrm{diag}(\mathbf{J}_{\mathbf{Fy}}^{\mathbf{i}})$$

$$\mathbf{J}_{Fy}^{i} = \begin{bmatrix} 0 & 0 \\ 0 & hE'_{q(i)}/(4H_{(i)}) \end{bmatrix}$$

$$\mathbf{J}_{gx} = \begin{bmatrix} \mathbf{J}_{gx(11)} & \mathbf{J}_{gx(12)} & \cdots & \mathbf{J}_{gx(1n)} \\ \mathbf{J}_{gx(21)} & \mathbf{J}_{gx(22)} & \cdots & \mathbf{J}_{gx(2n)} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{J}_{gx(n1)} & \mathbf{J}_{gx(n2)} & \cdots & \mathbf{J}_{gx(nn)} \end{bmatrix}$$

$$\mathbf{J}_{gx(ii)} = \begin{bmatrix} I_{d(i,\gamma)}\cos\delta_{(i,\gamma)} - I_{q(i,\gamma)}\sin\delta_{(i,\gamma)} + G_{(ii)}E'_{q(i)}\sin\delta_{(i,\gamma)} + B_{(ii)}E'_{q(i)}\cos\delta_{(i,\gamma)} & 0 \\ I_{d(i,\gamma)}\sin\delta_{(i,\gamma)} + I_{q(i,\gamma)}\cos\delta_{(i,\gamma)} + B_{(ii)}E'_{q(i)}\sin\delta_{(i,\gamma)} - G_{(ii)}E'_{q(i)}\cos\delta_{(i,\gamma)} & 0 \end{bmatrix}$$

$$\mathbf{J}_{gx(ik)} = \begin{bmatrix} G_{(ik)}E'_{q(k)}\sin\delta_{(k,\gamma)} + B_{(ik)}E'_{q(k)}\cos\delta_{(k,\gamma)} & 0 \\ B_{(ik)}E'_{q(k)}\sin\delta_{(k,\gamma)} - G_{(ik)}E'_{q(k)}\cos\delta_{(k,\gamma)} & 0 \end{bmatrix},$$

$$Jgy = \mathrm{diag}(J^{i}_{gy})$$

$$\mathbf{J}^{i}_{gy} = \begin{bmatrix} \sin\delta_{(i,\gamma)} & \cos\delta_{(i,\gamma)} \\ -\cos\delta_{(i,\gamma)} & \sin\delta_{(i,\gamma)} \end{bmatrix}$$

Similarly, one can find the analytical expression of Jacobian matrix when higher order machine model and dynamic load model are used. Among the reviewed the references, there should be no obvious difficulties to find the analytical expression of Jacobian matrix. Method to find Jacobian matrix in general case (rather than classic machine model and constant impedance) is described in [20]. In [26], analytical expression of Jacobian matrix is provided when transient machine model, dynamic exciter, and dynamic load model are used.

For the time stacking method, equations in (25) for several time steps are combined together to form a large set of algebraic equations:

$$\begin{cases} \mathbf{0} = \mathbf{H}(\mathbf{x}_\gamma, \mathbf{y}_\gamma) \\ \mathbf{0} = \mathbf{H}'(\mathbf{x}_\gamma, \mathbf{y}_\gamma, \mathbf{x}_{\gamma+1}, \mathbf{y}_{\gamma+1}) \\ \quad \cdots \\ \mathbf{0} = \mathbf{H}'(\mathbf{x}_{\gamma+p-2}, \mathbf{y}_{\gamma+p-2}, \mathbf{x}_{\gamma+p-1}, \mathbf{y}_{\gamma+p-1}) \end{cases} \quad (27)$$

where *p* represents the number of time steps in parallel, and **H** and **H'** comprise of equations similar to (24). In **H**, the previous time step variables are known and the only unknown are the current time step variables. In **H'**, variables of two future adjacent time steps are unknown. The variables to be determined are:

$$[(\mathbf{x}_\gamma, \mathbf{y}_\gamma), (\mathbf{x}_{\gamma+1}, \mathbf{y}_{\gamma+1}), \ldots, (\mathbf{x}_{\gamma+p-1}, \mathbf{y}_{\gamma+p-1})]$$

This equation set can be solved similarly as conventional one-step approach. The corresponding Jacobian matrix is

$$\mathbf{\Lambda} = \begin{bmatrix} \mathbf{J}_{\gamma} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \mathbf{J}_{(\gamma+1,\gamma)} & \mathbf{J}_{\gamma+1} & \dots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{J}_{(\gamma+p-1,\gamma+p-2)} & \mathbf{J}_{\gamma+p-1} \end{bmatrix} \quad (28)$$

where the diagonal sub-matrix is shown in (26). In the subscript of the off-diagonal matrix, the first number represents the equation set index, and the second number represents variable set index.

$$\mathbf{J}_{(\mathrm{j+1,j})} = \begin{bmatrix} \mathbf{J}_{F_{j+1}x_j} & \mathbf{J}_{F_{j+1}y_j} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \quad (29)$$

where j = ϒ, ϒ + 1, ..., ϒ + p-2. Each sub-matrix is calculated below:

$$\mathbf{J}_{F_{j+1}x_j} = \mathrm{diag}(\mathbf{J}^i_{F_{j+1}x_j})$$

$$\mathbf{J}^i_{F_{j+1}x_j} = \begin{bmatrix} -1 & -h\omega_b/2 \\ 0 & -1 + hD_{(i)}/(4H_{(i)}) \end{bmatrix}$$

$$\mathbf{J}_{F_{j+1}y_j} = \mathrm{diag}(\mathbf{J}^i_{F_{j+1}y_j})$$

$$\mathbf{J}^i_{F_{j+1}y_j} = \begin{bmatrix} 0 & 0 \\ 0 & hE'_{q(i)}/(4H_{(i)}) \end{bmatrix}$$

A prototype was successfully developed in MATLAB to implement the time-stacking method to prove the concept. This code considers reduced admittance matrix, constant impedance load model and classical generator model. The MATLAB scripts were also converted to FORTRAN code to implement parallel solvers for speedup purposes, which will be described in the following subsections. The detailed structure of the Jacobian matrix for the time-stacking method is shown in Figure 19 and Figure 20. A preliminary simulation was performed on the IEEE 16-generator-68-bus system model. Twenty steps were stacked together that can be solved simultaneously. The comparison of time-domain trajectories is made in Figure 21, which proves the concept of the proposed time-stacking method.

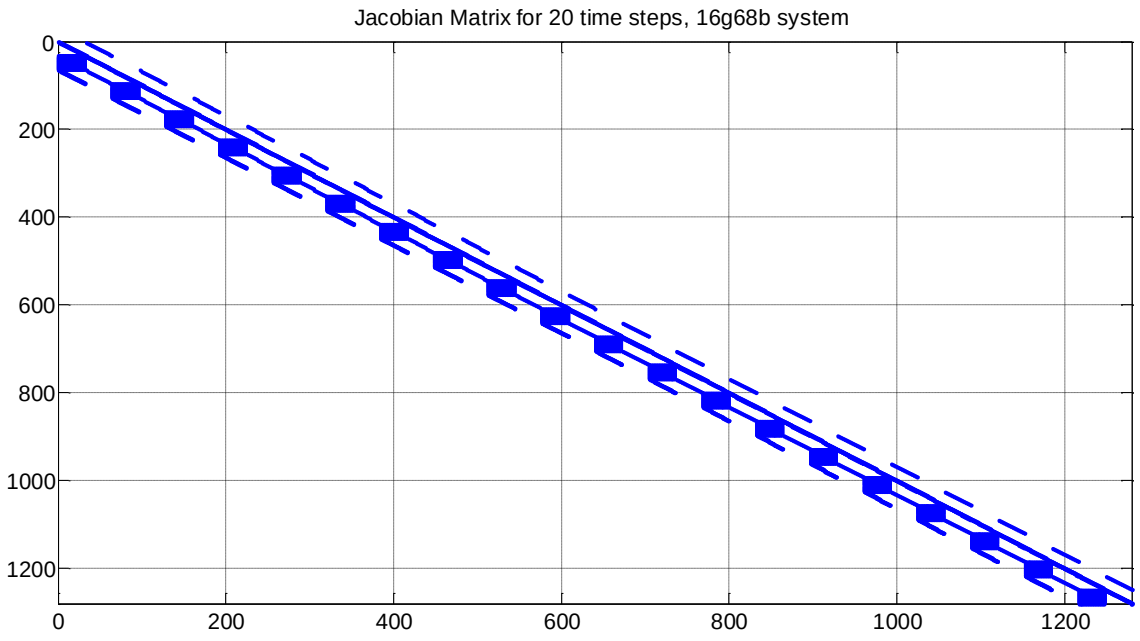


Figure 19. Structure of Jacobian matrix for the time stacking method (20 steps)

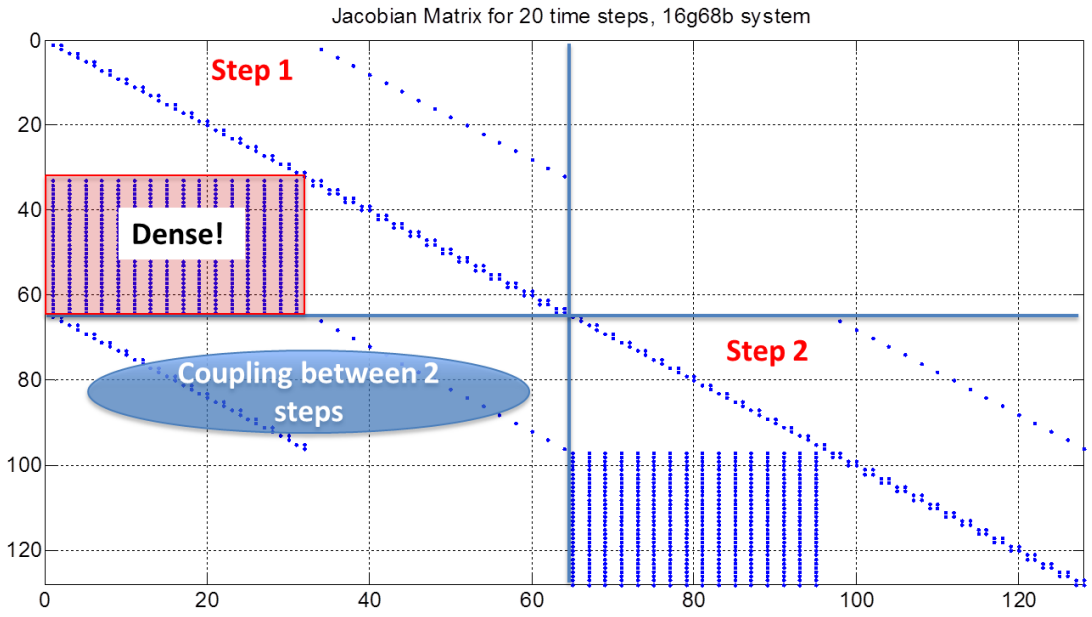


Figure 20. A zoom-in view of the Jacobian matrix elements with reduced admittance matrix

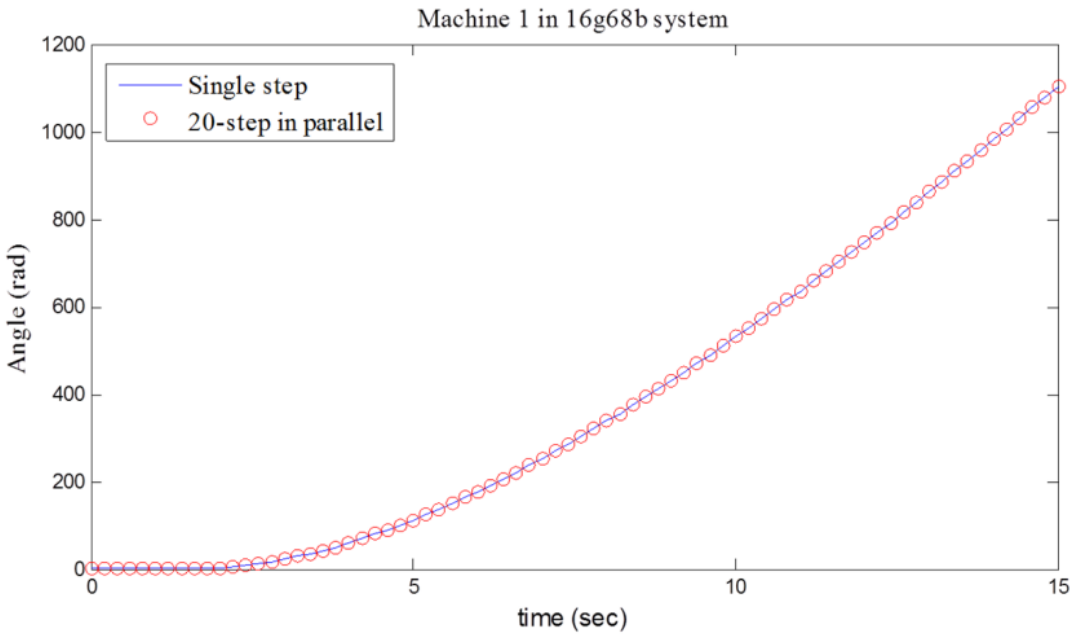


Figure 21. Proof of concept for the time-stacking method

4.5.2 Network admittance matrices (reduced vs. full)

As shown in Figure 19 and Figure 20, the Jacobian matrix for the time-stacking method contains a dense portion, if using reduced admittance matrix for network solution. The dense nature of the matrix might not be a desired feature in the trapezoidal method when solved in an iterative way, although the size is much smaller than the original, full admittance matrix (e.g., 3000 by 3000 vs. 16,000 by 16000 for a WECC size model). As suggested by the GE experts, PNNL team modified the formulation of the dynamic simulation by using full admittance matrix (larger but sparser than the previously used reduced admittance matrix). The complexity comparison is made between the reduced and full admittance matrices, shown in

Table 5. The corresponding MATLAB and FORTRAN codes were developed and tested to reflect this change. As expected, around 2 to 3 times of speedup were observed using the same simulation parameter settings and the computer hardware. Table 6 shows the comparison of computation time between the two schemes, for different test systems. It is observed that using the full admittance matrix can reduce the total simulation time using trapezoidal method from ~145s to ~54.2s. The SuperLU linear solver outperforms the DGESV solver. A preliminary scalability test is also performed to investigate the total computation time with respect to different number of processors, shown in Table 7 and Table 8. Around 15% of speedup is observed when using 4 processors are used to perform the dynamic simulation.

Table 5: Computational complexity comparison

per time step	Reduced Y	Full Y
Partitioned method with Modified Euler integration	Network: $O(n^2)$ Dense matrix vector multiplication and can be parallelized	Network: nnz of (LU) * iter No. Require forward and backward substitutions several times, and difficult to parallelize
	Others: O(n) or O(m)	
Simultaneous method	J: $8n^2$ multi + $4n^2$ sin (can be reduced to 2n) Y: $8n^2$ multiplication + $4n^2$ sin	J: O(m+n)
	Newton: one LU decomposition and a few substitution LU: ~$O(n^3)$	Newton: one LU decomposition and a few substitution LU:~$O((m+n)^2)$

m: number of buses; n: number of generators

Table 6: Computational time comparison

	Reduced Y-bus				Full-Ybus	
System	16g68b	50g145b	288g1081b	288g1081b	288g1081b	288g1081b
Linear solver	DGESV	DGESV	DGESV	SuperLU	SuperLU	SuperLU & Dishonest Newton
Admittance Matrix (s)	0.086	0.742	23.868	3.645	0.252	0.242
Jacobian (s)	0.121	0.792	25.139	4.299	0.378	0.128
Solver (s)	0.559	5.268	513.808	136.921	52.609	19.710
Total (s)	0.765	6.810	562.907	144.972	54.173	20.852

Table 7: Performance of parallel computing on the 288-generator-1081-bus model using Newton's method

	1 thread	2 threads	4 threads	8 threads	16 threads
y	0.252	0.244	0.203	0.176	0.165
Jac	0.378	0.396	0.267	0.292	0.301
superLU	52.609	48.851	45.974	51.676	99.023
Total (including others)	54.173	49.810	46.884	53.237	100.010

Table 8: Performance of parallel computing on the 288-generator-1081-bus model using Dishonest Newton's method

	1 thread	2 threads	4 threads	8 threads	16 threads
y	0.242	0.236	0.194	0.157	0.156
Jac	0.128	0.125	0.089	0.087	0.099
superLU	19.710	19.437	17.228	18.650	1.833
Total (including others)	20.852	20.615	18.105	19.461	36.842

4.5.3 Complexity of the time-stacking method

With the full admittance matrix and SuperLU solver, the time-stacking method was implemented in FORTRAN in the OPENMP environment to test the scalability of the time-stacking method. Table 9 summarizes the preliminary testing results for the 50-generator-145-bus model. The scalability of the time-stacking method is shown in Figure 22.

Table 9: Scalability testing of the time-stacking method with SuperLU and full admittance matrix

Stack 1 step	1 thread	2 threads	4 threads	8 threads
Admittance matrix	0.077	0.090	0.085	0.095
Jacobian matrix	0.087	0.148	0.110	0.145
Linear solver	9.800	9.410	8.490	8.950
Total (including others)	10.270	10.880	9.222	9.680
Stack 2 steps	1 thread	2 threads	4 threads	8 threads
Admittance matrix	0.064	0.075	0.078	0.088
Jacobian matrix	0.086	0.136	0.129	0.141
solver	13.393	11.747	11.327	12.667
Total (including others)	13.684	12.746	11.736	13.128
Stack 4 steps	1 thread	2 threads	4 threads	8 threads
Admittance matrix	0.055	0.060	0.055	0.057
Jacobian matrix	0.086	0.122	0.093	0.090
solver	19.327	18.542	14.351	15.330
Total (including others)	20.263	20.648	16.870	15.720

From this preliminary testing, the computation time of the time-stacking method is longer than the one for single time step implementation. This is expected due to the fact that additional coupling information is introduced when stacking multiple time steps together. For single step dynamic simulation, define the total computation time $\boldsymbol{T_s^{tot} = \frac{T_s}{u_s} t}$; for the time-stacking method, $\boldsymbol{T_p^{tot} = \frac{T_p}{u_p} * \frac{t}{p}}$, where t is the number of time steps, t_s is the average simulation time per set of equations at each single step using single thread, u_s is the speedup with multiple threads in single step method, t_p is the average simulation time per enlarged set of equations at each multi-step using single thread, and u_p is the speedup with multiple threads in time stacking method. In order to run the time-stacking simulation faster than single step simulation, we need to have $\boldsymbol{\frac{u_p}{u_s} > \frac{T_p}{pT_s}}$, which requires good scalability of a parallel linear solver.

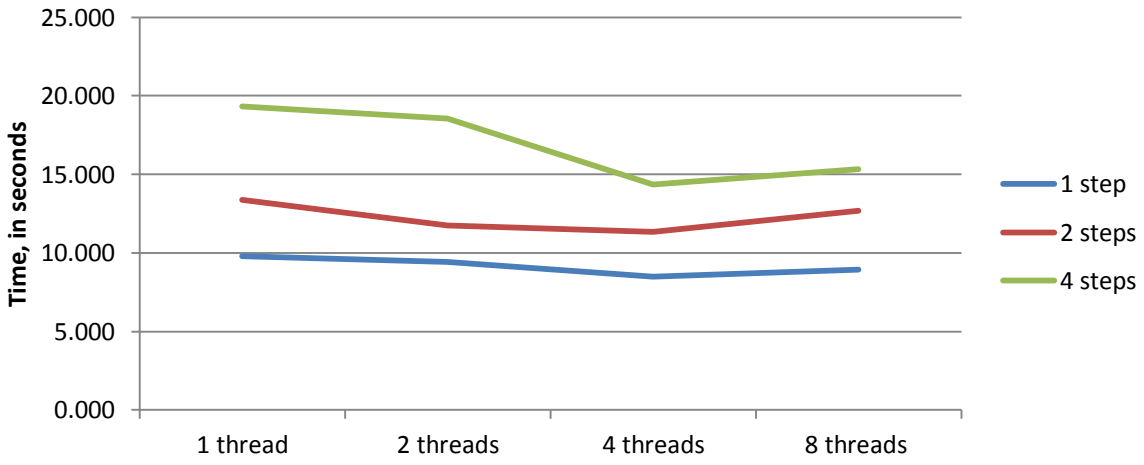


Figure 22. Scalability of the time-stacking method for different number of stacked steps with SuperLU solver

4.5.4 Performance of linear solvers

In order to speed up dynamic simulation, identification of a good linear solver is the key. PNNL team has investigated a number of linear solvers (direct and iterative solvers) that may achieve the best possible speed performance for the two target problems, (1) solving I=Y*U, for speeding up dynamic simulation in PSLF; and (2) solving $\Delta X = \text{Jacobian}^{-1} * \Delta Y$ for the proposed time-stacking method using trapezoidal integration.

4.5.4.1 Direct solvers for PSLF

It is found that for problem (2) and (3), direct solvers perform better than iterative solvers given the size (18,000 by 18,000 for a WECC system) and the property of the problem (re-factorization is only needed a couple times during a dynamic simulation). For the WECC size problem, we tested several direct solvers in MATLAB. It was found that the fastest direct solver can solve the problem much faster than the existing solver inside PSLF. After the selected solver is integrated into GE PSLF, the preliminary testing result shows more than 30 times of speedup to complete one linear solution. The total simulation time is reduced to half when using around 6 processors, with the same simulation settings. The simulation accuracy is also validated in PSLF.

4.5.4.2 Iterative solvers for the time-stacking method

For problem (4), iterative solvers perform equally well as direct solvers on a mid-size test system (16generator68bus with 20 steps stacked) once a good preconditioner is found. As an example, Figure 23 shows the structure of a Jacobian matrix (4,410 by 4,410 with 44,170 non-zeros) obtained in the time-stacking method, on a mid-size model. It shows a very large condition number: $7.9505*10^9$. To complete one solution for this Jacobian matrix, it takes 0.0182s using KLU solver. With the GMRes iterative solver and ILU as preconditioner, it takes 0.0784s to complete one solution using 1 processor. This iterative solver shows some scalability when using multiple processors: 0.0664s (2 processors) and 0.0365s (3 processors).

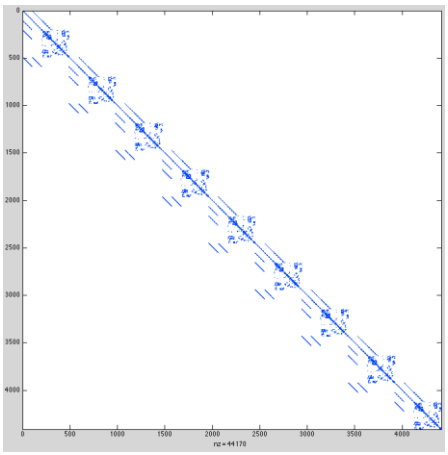


Figure 23 Structure of a Jacobian matrix for a mid-size system in the time-stacking method

For a larger problem with a bigger dimension in Jacobian, it is found that iterative solvers can outperform direct solvers using multiple processors. The problem size is increased to 44,100 by 44,100, when stacking more steps together. The testing results are compared in Figure 24. It clearly indicates that iterative solvers can beat KLU using 2 and more processors in solving this bigger linear problem. This is a desired feature for the time-stacking method on a large-scale model, indicating significant speedup can be expected for the time-stacking method. It is worth to mention that this iterative solver is only valid in MPI version, which is fundamentally different from the OPENMP parallel computing environment. The team is still in the process of identifying a good parallel linear solver that can be used on PNNL's FORTRAN scripts for the time-stacking method.

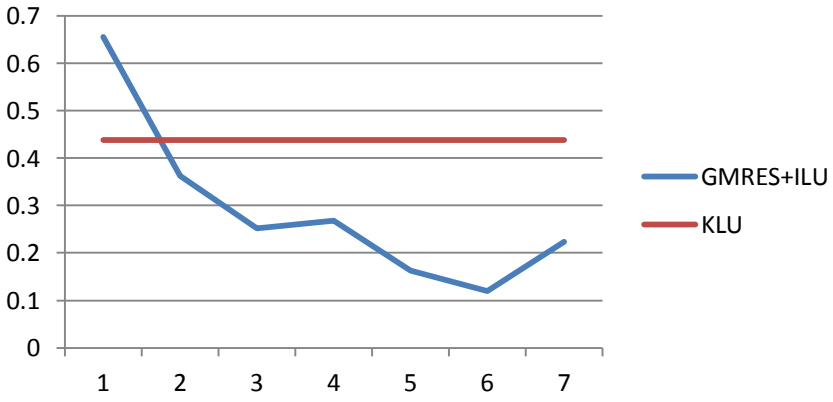


Figure 24. Performance comparison between GMRES and KLU solvers for solving a larger linear problem with a dimension of 44,100 * 44,100

4.5.5 Preliminary testing of adaptive time stepping method

PNNL team investigated several methods to vary time steps during the course of a dynamic simulation. The main objective is to further speed up dynamic simulation by using larger time steps when the power system is at quasi-steady state and using smaller time step at/near switching conditions. As an example, a smaller time step is used to simulate the detailed dynamic behavior of the system during a fault condition; and a larger time step can be used at initial conditions or after a disturbance is settled. It is found that up to 30% speedup can be observed using this technique without compromising simulation accuracy.

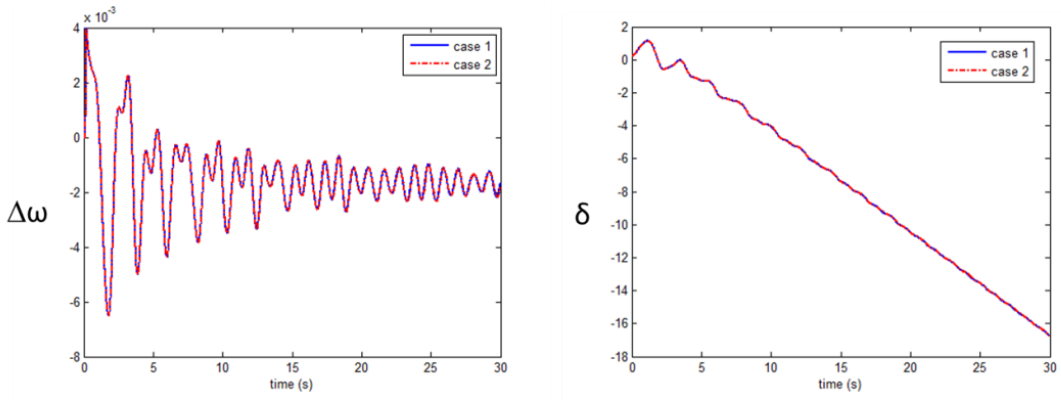


Figure 25. Accuracy of simulation results with adaptive time step (case 1: fixed time step; case 2: adaptive time step)

4.5.6 Implementation of detailed generator models with controllers

In order to obtain more realistic simulation results, state-space equations were developed for complex generator and controller models, including GENTPJ, EXAC2, and IEEEG1. The differential equations and their Jacobian matrices were developed from their control block, for the time-stacking method. Figure 26 through Figure 28 provide the control block diagrams for the 3 identified models. Limiters, saturations and nonlinear functions are all considered in Jacobian matrix. MATLAB codes were developed to implement these detailed models for dynamic simulations with the time-stacking method. The MATLAB code is then converted to C++ code for further testing purposes.

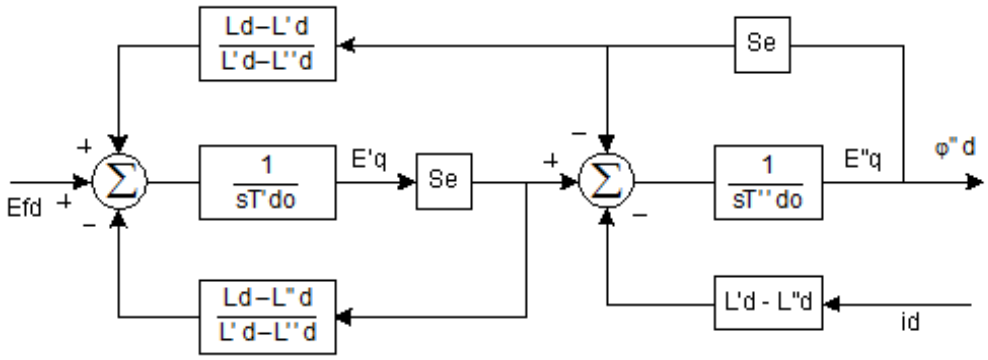


Figure 26. Control block diagram for GENTPJ (source: PSLF manual)

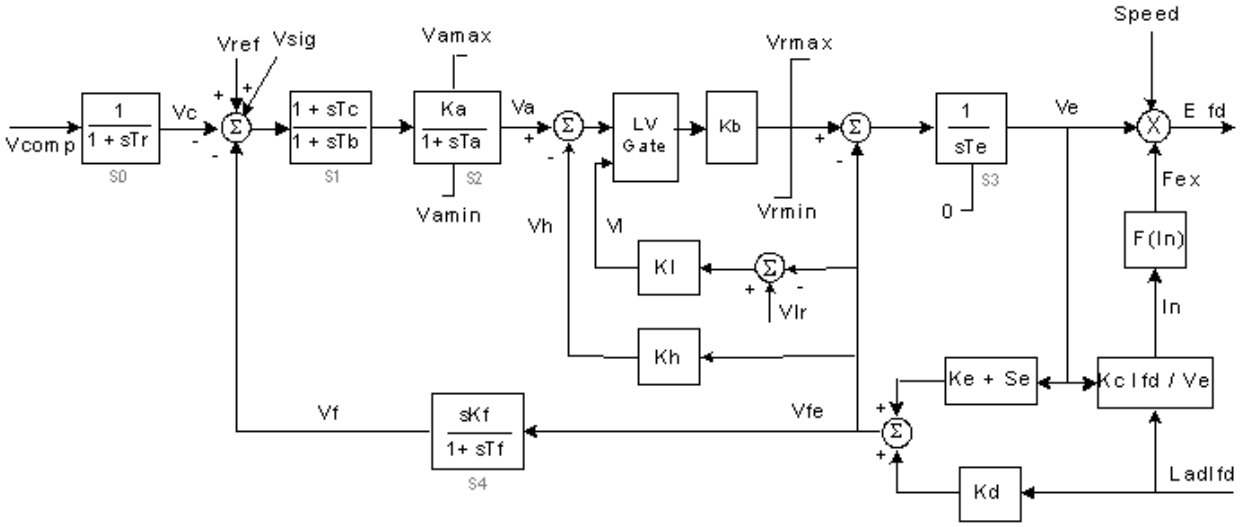


Figure 27. Control block diagram for EXAC2 (source: PSLF manual)

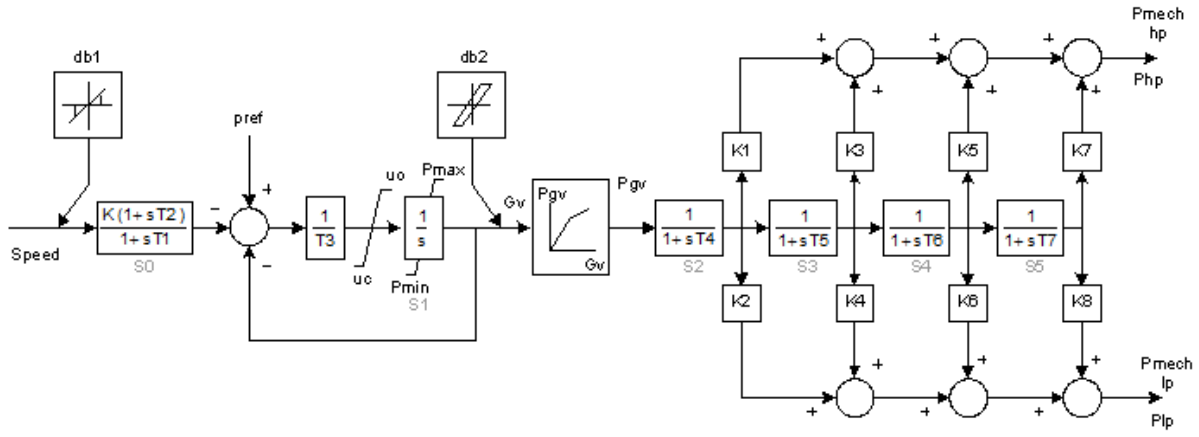


Figure 28. Control block diagram for IEEEG1 (source: PSLF manual)

4.6 Conclusions and Proposed Future Work

In the research project, several advanced computing techniques for speeding up a single dynamic simulation were explored and evaluated. A few major achievements were made, which include:

- PNNL team surveyed a few popular linear solvers in different platforms including both direct methods and iterative methods. A comprehensive study and comparison was made to identify the best for both GE PSLF software and the new time-stacking method for fast dynamic simulation. The best direct linear solver shows more than 30 times speedup compared to the existing solver in PSLF. This helps effectively enhance the speed of PSLF simulation (2x speedup using 6 processors on a WECC size model). For the time-stacking method, the best linear solver is GMRes iterative solver with ILU as preconditioner, which shows good scalability performance comparing to the single-core based KLU method.

- Methods/Prototypes developed
    - A simultaneous method (Trapezoidal rule)

      PNNL team implemented a dynamic simulation code in MATLAB (for proof of concept) and FORTRAN (for testing scalability in OPENMP environment) with the Trapezoidal method and classical generator models. We investigated and compared the computation complexity of using reduced (small but dense) and (big but sparse) full admittance matrices. Both Newton and dishonest Newton's methods were implemented.

    - A time stacking method

      A prototype was developed that implements the proposed time stacking method, in both MATLAB and FORTRAN language. The concept was proved. Detailed generator models and controllers (GENTPF, EXAC2, and IEEEG1) were developed to provide more realistic results.

    - An adaptive time stepping method

      PNNL team investigated several methods to vary time steps during the course of a dynamic simulation. The main objective is to further speed up dynamic simulation by using larger time steps when the power system is at quasi-steady state and using smaller time step at/near switching conditions. As an example, a smaller time step is used to simulate the detailed dynamic behavior of the system during a fault condition; and a larger time step can be used at initial conditions or after a disturbance is settled. It is found that up to 30% speedup can be observed using this technique.

From our efforts, the speed of running power system dynamic simulations can be significantly improved by parallel computing techniques.

As explained in earlier sections, a good parallel linear solver is the key to the success of the time-stacking method. The best possible parallel linear solver identified was only available in MPI version. The OPENMP version of the solver didn't show good scalability, indicating that future research is needed to either identify a better parallel iterative solver in OPENMP or reprogram the code for the time-stacking method in MPI environment to leverage the identified solver.

5 Contingency screening and ranking for small signal stability

Utilities today are in need of tools and techniques that will enable them to predict the dynamic stability and reliability of the grid in the real-time. The problem is challenging because of the large number of contingencies that are to be simulated. In this project a fast method for power system contingency screening and ranking for small signal stability assessment is presented which essentially reduces the number of contingencies for detailed evaluation. The proposed method avoids repeated computation of eigenvalues for all possible postcontingency scenarios. Instead, the eigenvalues corresponding to critical modes for post-outage conditions are estimated based on first-order eigenvalue-sensitivity using just the nominal condition eigenvalues and post outage system state matrices. Since a critical outage condition can produce a large change in the eigenvalues, the first order prediction might not have acceptable accuracy. To overcome this issue, a second-order correction is applied which needs the computation of the eigenvectors corresponding to the eigenvalues of interest. The proposed approach avoids the computationally expensive eigenvalue computation for each contingency case and helps to screen the harmful contingencies in real time.

5.1 Introduction

Poorly damped electro-mechanical modes remain to be of significant concern in modern power systems. Power system stabilizers (PSSs) have traditionally been employed to improve the damping of these modes. Although PSSs are effective in damping the local electromechanical modes, they are not quite effective for the inter-area modes. Therefore, these modes might become poorly damped or unstable in the worst case under certain operating conditions following contingencies. One of the goals of Dynamic Security Assessment (DSA) is to identify such critical contingencies and analyze the system security. Since a practical power system can have thousands of possible contingencies, it is not possible to analyze all of them using time-domain simulations. As a result, the first step in DSA is contingency screening and ranking which gives a list of cases that needs further attention. The focus of this paper is fast contingency screening and ranking for small-signal stability analysis of inter-area modes. Not much work has been reported in the area of contingency screening and ranking for DSA. Most of the research in this area focused primarily on the so-called 'first swing' stability of the power systems. Transient energy function methods involving the construction of a Lyapunov function following a contingency were used to evaluate the critical clearing time [27], [28], [29], [30], [31]. A composite index approach was proposed in [32] which used multiple indices to screen contingencies. Coherency based indices were proposed by Li et-al in [33] and this approach was compared with different transient energy function approaches in [34]. Eigenvalue sensitivity was used in [35] for computing modal synchronizing torque coefficient for contingency screening for the first swing stability. The power transfer distribution factor (PTDF), which is calculated based on a dc power flow assumption has been used in literature [36], [37] to determine the post contingency operating condition. In [38], the PTDF has been used for contingency screening and ranking. This approach can avoid the computational burden of the full AC load flow and determine the post outage operating condition using DC load flow. The trade-off, however, is the accuracy of the post contingency condition. It was shown in [38] that this approach was unable to capture almost half of the unstable contingencies in a 3-machine WSCC system [24] and 80% of the unstable contingencies in the 10-machine 39-bus New England test system [39].

In [40], contingency screening was performed for extremely large contingencies like (N - 2) and (N - 3) based on eigenvalue sensitivity to small perturbations in line admittance and shunt capacitor. This is an approximation to the effect of line outage, which is a large disturbance. Moreover, the contingencies were sorted based on the magnitude of eigenvalue sensitivity -the higher the magnitude, the greater is the criticality of the contingency. This criteria does not indicate how critical the modes are i.e., how far they are from the imaginary axis. In the proposed approach of this paper, we do not consider a dc load flow approximation. The eigenvalues for a post-outage scenario are estimated from a system state matrix which is computed after the removal of the line rather than a small change in its admittance. Moreover, the proposed algorithm considers the 2nd order approximation in order to take into account the non-linearity due to the large change in the system matrix resulting from the line outage. In the proposed approach, the contingencies are sorted based on estimation of eigenvalues corresponding to inter-area modes. The method focuses on eigenvalues associated with the critical modes rather than any indirect way such as screening based on eigenvalue sensitivity.

5.2 Eigenvalue sensitivity based fast contingency screening

Power system dynamic behavior is typically modeled using a set of differential-algebraic equations (DAEs)

$$\begin{aligned} \dot{x} &= f(x, y) \\ 0 &= g(x, y) \end{aligned} \tag{30}$$

where, x is a vector of state variables associated with the dynamic states of generators, loads, and other system components; y is a vector of algebraic variables associated with steady-state variables such as voltage phasor magnitudes and angles. Small signal stability analysis is done by linearizing these non-linear equations around an operating point $(\mathrm{x}_0, \mathrm{y}_0)$

$$\begin{bmatrix} \Delta\dot{x} \\ 0 \end{bmatrix} = \underbrace{\begin{bmatrix} J_1 & J_2 \\ J_3 & J_4 \end{bmatrix}}_{J} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \tag{31}$$

where, J is the system Jacobian matrix, and $J_1 = \partial f/\partial x|_0$, $J_2 = \partial f/\partial y|_0$, $J_3 = \partial g/\partial x|_0$, $J_4 = \partial g/\partial y|_0$ are the respective partial derivatives evaluated at the operating point. The system eigenvalues can be computed by eliminating the vector of algebraic variables, so that the DAE system is reduced to a set of purely ordinary differential equations (ODE)

$$\Delta\dot{x} = (J_1 - J_2 J_4^{-1} J_3)\Delta x = A_{sys}\Delta x \tag{32}$$

The eigenvalues and eigenvectors of A_{sys} are given by

$$A_{sys}\phi_i = \lambda_i \phi_i \tag{33}$$

$$\varphi_i A_{sys} = \lambda_i \varphi_i \tag{34}$$

where, λ_i, ϕ_i and φ_i are the eigenvalue, right eigenvector and left eigenvector respectively corresponding to mode i. The eigenvalues of the above ODE system provide all the information on modal damping ratios and frequencies. For each mode, characterized by a pair of complex conjugate eigenvalues, the corresponding right eigenvector indicates the mode shape, giving information on the relative phase of each state in that particular mode. The participation factors provide information on the influence of states on modes, and can be computed using the left and right eigenvectors.

The eigenvalues of the above linear system provide the information on modal damping ratios and frequencies. Small-signal stability assessment is the small-signal stability analysis of the system under a set of contingencies for a range of operating conditions. The system is considered to be small-signal secure if the damping ratio/settling time of all the oscillatory modes is within a required threshold value. For the problem of inter-area oscillations, we focus on modes in the frequency range of 0.1 Hz to 1 Hz. Oscillatory modes with damping ratio less than a threshold value, such as 0.05, are considered to be the critical modes of concern. The system state matrix A_{sys} changes with changes in the operating point, since it is formed based on a linearization around an operating point. The operating point changes as a result of a contingency such as loss of a transmission line due to a fault. In order to analyze the impact of contingencies on small signal stability of the system, one approach is to re-compute the eigenvalues for post-contingency situations by solving (4). Upon solving this equation for each case, the contingencies can be ranked according to their impact on the eigenvalues. The contingencies that result in the maximum movement of the eigenvalue towards the positive side of the real axis are considered as the critical contingencies.

For a large system, solving (33) for each post-contingency condition is a computationally challenging problem. This problem becomes infeasible when the goal is to perform such a contingency analysis in near-real-time. Another approach is to perform time domain simulations for all the contingency conditions, and evaluate the change in damping of critical modes using measurement-based modal identification techniques. Here again, simulating large number of contingencies in time domain for a large-scale system is a time consuming process, and infeasible for near-real-time applications.

The goal of the proposed algorithm is to screen the critical contingencies for a given poorly damped mode without either solving (33) for each contingency or simulating the computationally intensive time-domain dynamic simulations. The algorithm is based on an eigenvalue sensitivity-based approach. In this approach, the eigenvalue corresponding to the given critical mode is *estimated* based on the eigenvalues and eigenvectors at the nominal condition and change in the system matrix for each contingency. The advantage of the proposed algorithm is that we avoid the computationally challenging task of solving (33) for each contingency.

Eigenvalue sensitivity is a very useful tool to understand the small-signal stability problem, and has been used in various applications in the past. Here, we consider its use in analyzing the impact of contingencies, which result in changes in operating point. The 1st order eigenvalue sensitivity is given by

$$\frac{\partial \lambda_i}{\partial \gamma} = \frac{\psi_i \frac{\partial A}{\partial \gamma} \phi_i}{\psi_i \phi_i} \quad (35)$$

where, γ is any system parameter. The above equation gives information on sensitivity of the eigenvalue corresponding to mode i for a small change in the system parameter γ. Both left and right eigenvectors are used in the calculation. Multiplying both sides of above equation by $\Delta\gamma$, we get

$$\frac{\partial \lambda_i}{\partial \gamma} \Delta\gamma \cong \frac{\psi_i \Delta A \phi_i}{\psi_i \phi_i}$$

where, $\Delta A = (A_{post} - A)$ is the change in system state matrix from nominal condition to a post-contingency condition. Using Taylor series approximation, the eigenvalue for the post-contingency condition can be estimated by

$$\lambda_{post} \cong \lambda_i + \frac{\partial \lambda_i}{\partial \gamma} \Delta\gamma \cong \lambda_i + \frac{\psi_i \Delta A \phi_i}{\psi_i \phi_i} \varphi_i \quad (36)$$

The only term in (36) dependent on the post-contingency condition is ΔA. All the other terms are computed only once – at the nominal condition. Since ΔA is computed using nominal condition and post-contingency condition, the change in operating condition due to a contingency is taken into account in the estimation algorithm, without needing to re-compute the eigenvalue using (33). So, although the estimation of eigenvalue is based on a 1st order approximation, the post-contingency condition is accounted for through the ΔA term.

While (36) takes into account the change in operating condition as a result of a contingency, it is bound to have certain inaccuracy in predicting the eigenvalue due to the underlying 1st order approximation. To improve the accuracy in this regard, we include the 2nd order term. Then, the estimate of eigenvalue is given by

$$\lambda_{post} \cong \lambda_i + \frac{\partial \lambda_i}{\partial \gamma} \Delta\gamma + \frac{1}{2!} \frac{\partial^2 \lambda_i}{\partial \gamma^2} (\Delta\gamma)^2$$

$$\cong \lambda_i + \frac{\psi_i \Delta A \phi_i}{\psi_i \phi_i} + \frac{1}{\psi_i \phi_i} \left[\psi_i \Delta A \sum_{\substack{k=1 \\ \neq i}}^{n} \left(\frac{\psi_k \Delta A \phi_i \phi_k}{\psi_k \phi_k (\lambda_i - \lambda_k)} \right) \right] \quad (37)$$

The second order term in (37) uses information on all the eigenvalues, left and right eigenvectors computed at the nominal condition in order to estimate the eigenvalue for mode i. The 1st order estimation in (36), however, requires only the eigenvalue and eigenvector corresponding to that particular mode. Hence, there is a tradeoff between accuracy and computation when the eigenvalue is estimated using a higher order approximation vs. a first order approximation.

5.3 Feasibility of Practical Implementation

The steps followed in determining the critical eigenvalues under post-contingency operating scenarios as described in the previous sub-section are summarized in the flow chart shown in Figure 29. The three key components of the proposed estimation method are highlighted in Blocks I, II and III. Following the completion of these key steps, the contingencies are screened and ranked based on the **real-part of the estimated eigenvalues**. The real-part of the eigenvalue is used as the indicator of how far the eigenvalue is from the imaginary axis. A contingency is classified as 'critical' if the real-part of eigenvalue is greater than a threshold value (e.g., -0.133 corresponding to a settling time of 30 seconds). After screening and ranking the critical contingencies, time-domain simulations can be run to validate the results for the most critical contingencies. Time-domain simulations are run using HPC platform to validate the results for the most critical contingencies.

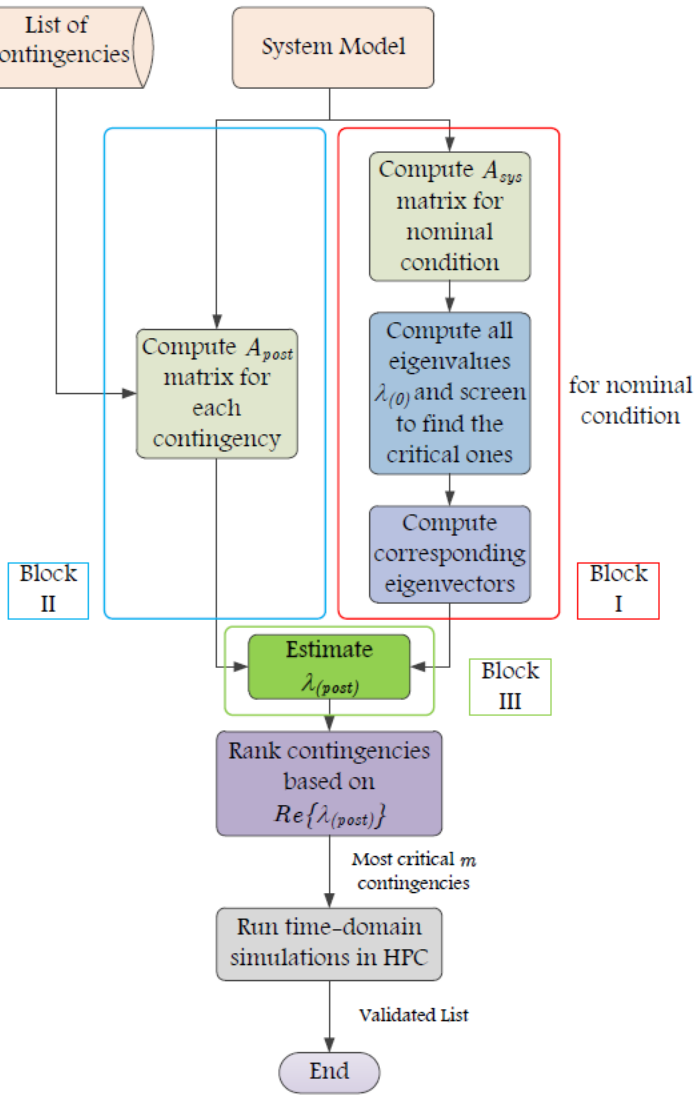


Figure 29: Flowchart of the proposed contingency screening algorithm.

As mentioned earlier, the eigenvalues for the post-contingency scenarios can be calculated by solving (33). In this report the '*eig*' function of MATLAB, which utilizes the LAPACK library from Intel Math Kernel Library (Intel MKL) is used for computing the eigenvalues from the full state matrix. This is a computationally efficient way of calculating the eigenvalues using commercial grade software such as MATLAB. This method will be referred to as 'EIG' in the rest of the report.

Blocks I and II are common in the EIG approach and the proposed approach. In the proposed approach, Block III estimates only a few selected eigenvalues which correspond to the critical electro-mechanical modes.

While the proposed eigenvalue sensitivity-based algorithm gets rid of the computationally challenging task of solving (33) for each contingency, it still has portions that may pose challenges when considering real-time implementation on large-scale systems. One of the challenges is in the computation of the 2nd order term in Block III. As can be seen from (37), this term requires the knowledge of all the eigenvalues and left and right eigenvectors at the nominal condition, which could be a challenging problem for a large-scale system. Furthermore, in Block II, the system state matrix has to be computed for all the contingency conditions, and this could also be a challenging problem for a large-scale system.

Two approaches have been identified in this task to further improve the computational efficiency of the proposed algorithm. The two approaches are described below.

5.3.1 Approach 1

5.3.1.1 Improving speed of computation of Block III:

For an efficient calculation of Block III, the matrix multiplication and addition in (36) and (37) should be done judiciously. This computation can be made more efficient as follows. The left and right eigen-matrices and the eigenvalues are computed from Block I, and can be arranged in matrix form as,

$$\Phi = \begin{bmatrix} \phi_1^T \\ \phi_2^T \\ \vdots \\ \phi_n^T \end{bmatrix}_{(n\times n)} \qquad \Psi = \begin{bmatrix} \psi_1^T \\ \psi_2^T \\ \vdots \\ \psi_n^T \end{bmatrix}_{(n\times n)} \qquad \Lambda = \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_n \end{bmatrix}_{(n\times n)}$$

Rows and columns corresponding to the critical eigenvalues are selected from these matrices. Let us assume $\lambda_1, \lambda_2, \dots, \lambda_{ncrit}$ are the critical eigenvalues of interest.

1) First order estimate: The estimation of the critical eigenvalues for a particular post-contingency scenario is described in (36). To calculate the first order term, the following arrangement of matrix multiplication leads to an efficient computation:

$$\Delta\Lambda = \begin{bmatrix} \Delta\lambda_1 \\ \Delta\lambda_2 \\ \vdots \\ \Delta\lambda_i \\ \vdots \\ \Delta\lambda_{ncrit} \end{bmatrix} = diag\left\{ \begin{bmatrix} \psi_1 \\ \psi_2 \\ \vdots \\ \psi_i \\ \vdots \\ \psi_{ncrit} \end{bmatrix} [\Delta A_{sys}] \begin{bmatrix} \phi_1^T \\ \phi_2^T \\ \vdots \\ \phi_i^T \\ \vdots \\ \phi_{ncrit}^T \end{bmatrix} \right\}$$

2) Second order estimate: Similarly, for the second order term, the summation term can be calculated efficiently in the following steps of matrix multiplications and additions:

Step I: the following expression is computed.

$$\Gamma = \left[\Psi \Delta A_{sys} \begin{bmatrix} \phi_1^T \\ \phi_2^T \\ \vdots \\ \phi_r^T \end{bmatrix}_{(n \times ncrit)} \right]$$

where, $\Gamma = [\Gamma_1 \quad \Gamma_2 \quad \cdots \quad \Gamma_{ncrit}]$

Step II: corresponding to the *i*th critical eigenvalue $\Delta\lambda_{ik}$ is computed as

$$\Delta\lambda_{ik} = [(\lambda_i - \lambda_1) \quad (\lambda_i - \lambda_2) \quad \cdots \quad (\lambda_i - \lambda_n)]^T$$

Step III: the matrix given below is evaluated:

$$\Xi^{\mathrm{i}} = \Gamma_i \cdot / \Delta\lambda_{ik}$$

where, the dot operator represents element-wise division in matrices.

StepIV: The *i*th row of the above matrix Ξ^i and the *i*th column of right eigenvector ϕ are removed and the corresponding variables are defined as $\Xi^{[i]}$ and $\phi^{[i]}$ respectively.

Step V: the summation term in (8) is computed by summing over the columns of the matrix $(\phi^{[i]} \text{ x } \text{diag}\{\Xi^{[i]}\})$ given by: $sum\{col(\phi^{[i]} \times diag\{\Xi^{[i]}\})\}$

Step VI: steps II to V are repeated for $i = 1, 2, \ldots, ncrit$.

5.3.1.2 Improving speed of computation of Block II:

Calculation of A_{sys} consumes a significant portion of time since it needs to be performed for all post contingency conditions. One popular numerical method to compute A_{sys} is the perturbation based technique, where each state is perturbed individually. This computation can be reduced by *selectively* perturbing the states that have higher impact on a given critical mode. This information is obtained

by computing the participation factors for a given mode. Participation factors of the states for a particular mode are computed using the left and right eigenvectors, the sum of which equals 1 for a given mode. The states that contribute to a majority of participation in a given mode, given by the cumulative sum of participation factors can be chosen as the important states. This process can potentially improve the speed of computation of sys matrix for each contingency. As will be demonstrated in the next section, the threshold value used to determine the majority contribution (e.g., cumulative sum equal to 0.95) dictates the accuracy of the algorithm. Smaller the threshold, lesser the number of states that need to be perturbed in order to compute the A_{sys} matrix, but greater is the inaccuracy in estimating the eigenvalues.

The shortcoming of this approach is that it requires the computation of the full eigenspace in order to calculate the participation factors. This may be impractical for a large scale system, especially when considering a dense system state matrix for real-time application.

5.3.2 Approach 2

To overcome the short coming of approach 1, we consider an alternate way to compute the second order term in the eigenvalue estimation. In the second order term in (37), we can see that there is a summation term that requires all the eigenvalues and eigenvectors at the operating point. This can consume significant computation time. We recall (37) below.

$$\lambda_{post} \cong \lambda_i + \frac{\partial \lambda_i}{\partial \gamma} \Delta\gamma + \frac{1}{2!}\frac{\partial^2 \lambda_i}{\partial \gamma^2}(\Delta\gamma)^2$$

$$\cong \lambda_i + \frac{\psi_i \Delta A \phi_i}{\psi_i \phi_i} + \frac{1}{\psi_i \phi_i}\left[\psi_i \Delta A \sum_{\substack{k=1 \\ \neq i}}^{n}\left(\frac{\psi_k \Delta A \phi_i \phi_k}{\psi_k \phi_k (\lambda_i - \lambda_k)}\right)\right]$$

The second order term in the above expression can be written alternatively as

$$\frac{\partial^2 \lambda_i}{\partial \gamma^2}(\Delta\gamma)^2 \cong \frac{1}{\psi_i \phi_i}\left[2\psi_i\left\{\frac{\partial A_{sys}}{\partial \gamma}\Delta\gamma - \frac{\partial \lambda_i}{\partial \gamma}(\Delta\gamma) I\right\}\frac{\partial \phi_i}{\partial \gamma}\Delta\gamma\right]$$

$$\cong \frac{1}{\psi_i \phi_i}\left[2\psi_i\left\{\Delta A_{sys} - \Delta\lambda_i^{1st} I\right\}\frac{\partial \phi_i}{\partial \gamma}\Delta\gamma\right] \quad (38)$$

Multiplying on both sides of the above equation by $\Delta\gamma$, we get

$$\begin{bmatrix} A_{sys} - \mathrm{Re}(\lambda_i I) & \mathrm{Im}(\lambda_i I) \\ -\mathrm{Im}(\lambda_i I) & A_{sys} - \mathrm{Re}(\lambda_i I) \\ \mathrm{Re}(\phi_i^*) & \mathrm{Im}(\phi_i^*) \end{bmatrix}\begin{bmatrix} \mathrm{Re}\left(\frac{\partial \phi_i}{\partial \gamma}\right)\Delta\gamma \\ \mathrm{Im}\left(\frac{\partial \phi_i}{\partial \gamma}\right)\Delta\gamma \end{bmatrix} = \begin{bmatrix} -\mathrm{Re}(\Delta A_{sys} - \Delta\lambda_i^{1st} I)\phi_i \\ -\mathrm{Im}(\Delta A_{sys} - \Delta\lambda_i^{1st} I)\phi_i \\ 0 \end{bmatrix} \quad (39)$$

The above equation is in the form $Ax = b$. In this equation, all the terms are known except the b term. The remaining terms are dependent on just the system state matrices for nominal and post-

contingency conditions, the eigenvalues and eigenvectors corresponding to a given electro-mechanical mode, and the first order estimate. Solving this equation, and substituting the resulting eigenvector sensitivity in the second order estimate equation (38), one can obtain the second order term. This approach has the advantage that the computation of the full eigenspace is avoided. Only the eigenvalues and right eigenvectors corresponding to the modes of interest need to be known. There exist computationally efficient techniques that can perform such a selected eigenspace computation for large-scale dense matrices.

Other approaches to improve computational efficiency:

Exploiting sparsity in full DAE formulation: The proposed eigenvalue sensitivity-based algorithm uses the system state matrix A_{sys}, which is dense. Instead, the sparse nature of the full DAE model given by (31) can be exploited. It is straightforward to extend the proposed algorithm to the full DAE formulation. This extension to the full DAE formulation could potentially benefit the contingency screening algorithm.

Application of High Performance Computing (HPC): The proposed algorithm requires the computation of system state matrices corresponding to all the contingencies. This process is easily parallelizable, and so HPC can be used to speed up this computation. Computation of A matrix involves perturbing each of the dynamic states and computing the resulting changes in system states and this process is inherently parallelizable. Parallel computing techniques can be exploited to allocate tasks that are performed in a loop to multiple cores. Also, the analysis for one contingency is independent of the others. Therefore, the algorithm can be easily parallelized in order to analyze multiple contingencies. Such a parallelization can lead to a significant reduction in the computational time of the overall algorithm.

5.4 Results

5.4.1 16-machine 68-bus system

A 16-machine 68-bus test system is used to demonstrate the effectiveness of the proposed contingency screening algorithm. The description of this test system is included in Appendix A and the detail small signal characteristics is described in Appendix B. As shown in Table 25 in Appendix B, this test system has 4 inter-area modes. For the purpose of this analysis, we will focus on the Mode 1 (~ 0.37 Hz mode).

For (N-1) contingency analysis in this system, 87 feasible line outage scenarios have to be considered. The goal of the proposed algorithm is to reduce the number of contingencies to be analyzed. Table 10 summarizes the results of this algorithm. Each row in the table corresponds to a contingency. Note that here we do not show the results for all possible contingencies, but for only a selected set, although the algorithm is run for all possible contingencies. The table lists the estimated real part of eigenvalue corresponding to a particular critical mode using the second order based approach as well as the first order based approach. It also lists the "benchmark" results that can be obtained using exact eigenvalue computation. As can be seen from the table, across all the selected contingencies the second order based approach performs better (lower error) than the first order based approach. The error shown here is simple the absolute value of the difference between

estimated value and benchmark value. By setting a threshold value for the real part of eigenvalue of -0.1 (corresponding to a settling time of 40 seconds), we can see that the top 6 contingencies (highlighted in blue) are critical based on the benchmark results. Since the proposed algorithm estimates eigenvalues, we added a 10% margin to this threshold in order to arrive at the list of critical contingencies. The resulting list of critical contingencies based on the second order estimation is highlighted in red. As can be seen, all the critical contingencies obtained based on the actual computation are captured by the algorithm. The algorithm also identifies 2 additional contingencies, due to the added margin. This demonstrates that the proposed algorithm is successful in identifying the critical contingencies. The table also lists a ranked list of contingencies for the first order and second order based approaches along with the benchmark ranking list. As can be seen from this, the second order based approach is able to capture the top ten contingencies and maintain most of the ranking. However, the first order based approach is unable to capture contingencies ranked 9 and 10. This demonstrates the importance of the second order term in accurately estimating the eigenvalues post contingency and capturing the critical contingencies.

Table 10: Comparison of estimated and actual eigenvalues for contingency screening and ranking (Base Case with default load model)

Line		Real part	Real part	Real part	Error	Error	Ranking	Ranking	Ranking
From	To	(2nd order)	(1st order)	(EIG)	(2nd order)	(1st order)	(2nd order)	(1st order)	(EIG)
37	68	-0.0517	-0.0865	0.0129	0.0646	0.0994	1	1	1
22	21	-0.0805	-0.1024	-0.0354	0.0451	0.0670	3	4	2
47	53	-0.0782	-0.0948	-0.0755	0.0026	0.0193	2	2	3
21	68	-0.1031	-0.1124	-0.0899	0.0132	0.0225	6	8	4
27	37	-0.1000	-0.1044	-0.0980	0.0019	0.0064	4	5	5
40	48	-0.1008	-0.0952	-0.1002	0.0006	0.0050	5	3	6
61	60	-0.1061	-0.1133	-0.1029	0.0032	0.0104	7	9	7
61	60	-0.1061	-0.1133	-0.1029	0.0032	0.0104	8	10	8
54	53	-0.1103	-0.1164	-0.1076	0.0027	0.0088	9	16	9
54	53	-0.1103	-0.1164	-0.1076	0.0027	0.0088	10	17	10

A key aspect of the proposed approach is the computation time of the algorithm. Table 11 below shows the computation times for a single contingency using the 1st order approach, 2nd order approach and the EIG approach. As can be seen from the table, the 1st order approach takes the least amount of time, followed by the 2nd order approach and then the EIG approach. The 1st order approach is almost 27 times 5 faster than the EIG approach, and the 2nd order approach is 2:5 times faster than the EIG approach. The computer used to perform these simulation had the following configuration: IntelR XeonR CPU W3565 @ 3.2 Ghz, 4 Cores, 8GB RAM, Windows 7 Professional, 64-bit, and the MATLAB version used was R2013b. PSLF solution architecture.

Table 11: Comparison of computation times in second

(1st order)	(2nd order)	EIG
0.000433	0.00465	0.011654

Therefore by first screening the contingencies with first order approach considering higher threshold on real-part of eigenvalue (damping or settling time) and then computing the second order terms for only those contingencies that exceeds the threshold, a significant reduction in computation time can be achieved.

Figure 30 shows the graphical representation of the two oscillating generator group for 0.37 Hz mode. Group 1 generators are shown in green and Group 2 generators are shown in red. The circle represents the generator and magnitude and angle of the arrow represents the modehape magnitude and angle for 0.37 mode. The critical contingences that cause this mode to be unstable/lightly damped are shown in blue colored line.

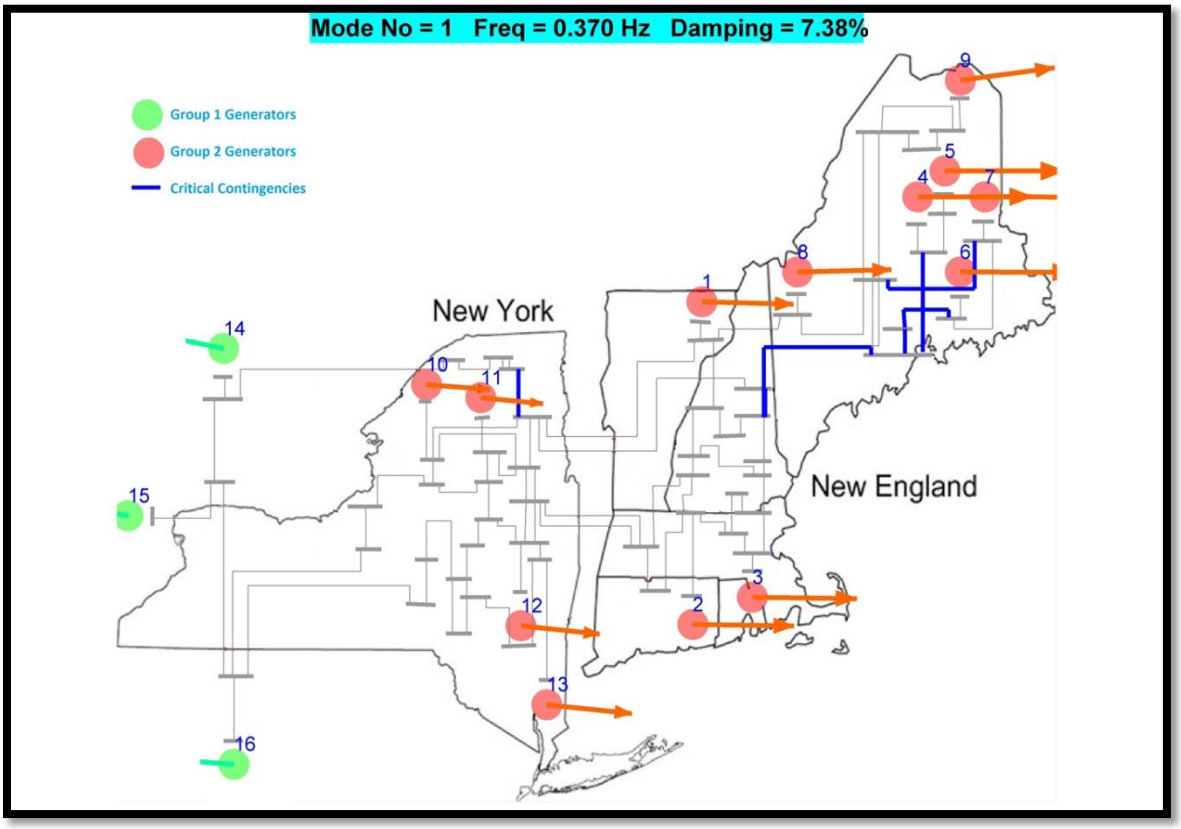


Figure 30: Graphical representation of oscillation group and critical contingences for 16 machine 68 bus system.

Table 12 summarizes the results of this algorithm for the same 16 machine test system, but at a different operating condition considering constant impedance load model. This case was developed to represents a lightly damped system which can then be used to demonstrate the effectiveness of optimal control action. The control action necessary to bring the system back to stability following the critical contingencies is explained using this condition in next section. As can be seen from the table, similar to the previous test condition, here as well the 2nd order based approach provides better results than the 1st order based approach. This can be concluded by looking at the error values for each of the top 10 contingencies for 2nd order and 1st approaches in table 13.

Table 12: Comparison of estimated and actual eigenvalues for contingency screening and ranking (Lightly damped case with constant impedance load model)

Line		Real part	Real part	Real part	Error	Error	Ranking	Ranking	Ranking
From	To	(2nd order)	(1st order)	(EIG)	(2nd order)	(1st order)	(2nd order)	(1st order)	(EIG)
37	68	0.1403	0.0542	0.2270	0.0867	0.1728	1	1	1
22	21	0.0622	0.0040	0.1372	0.0750	0.1332	2	2	2
21	68	-0.0050	-0.0336	0.0235	0.0285	0.0571	3	6	3
61	60	-0.0080	-0.0234	-0.0078	0.0003	0.0156	4	3	4
61	60	-0.0080	-0.0234	-0.0078	0.0003	0.0156	5	4	5
27	37	-0.0182	-0.0288	-0.0129	0.0053	0.0159	6	5	6
54	53	-0.0222	-0.0358	-0.0217	0.0005	0.0141	7	7	7
54	53	-0.0222	-0.0358	-0.0217	0.0005	0.0141	8	8	8
24	23	-0.0300	-0.0444	-0.0237	0.0063	0.0207	9	9	9
63	58	-0.0356	-0.0489	-0.0302	0.0054	0.0188	10	10	10

5.5 Conclusions

A novel eigenvalue sensitivity-based approach for screening contingencies for small signal stability has been proposed and tested on the 16-machine 68-bus New York-New England Test. The result shows that the proposed approach of estimating modes for post contingency condition is suitable in identifying critical contingencies with significant speed gain. While the second order correction entails more computation burden than the first order, it can more accurately screen and rank the contingencies in the order of severity. With the improved availability of parallel computing techniques for utility operations, the proposed approach has the potential to enable real time decision making to predict and tackle oscillatory problems based on current operating conditions.

6 Oscillation damping control

6.1 Introduction

The objective of this task is to develop an algorithm that will determine the optimum generation and/or load change required to improve the damping of poorly damped oscillatory modes in the system for all N-1 contingencies. The proposed algorithm takes advantage of a model-based approach to improve the oscillatory stability of the system considering "what-if" or N-1 contingency conditions. The broad idea is to leverage the fast time domain simulation engine to estimate the sensitivity of eigenvalues with respect to tunable system parameters (e.g., generation re-dispatch) and use this information to determine the required control action scheme that the system operator can take. One key benefit of adopting an eigenvalue sensitivity based approach is that the same information can be leveraged for the contingency screening algorithm proposed earlier. For example, if the system operator needs to analyze the impact of generator outage contingencies on system oscillatory modes, the eigenvalue sensitivity can be used to obtain such information. As a byproduct of that, the sensitivity can be used in the control algorithm proposed in this task.

In any generation re-dispatch based oscillation damping control scheme, there are four basic questions that need to be answered:

1. Which generators should increase their output and which ones should decrease their output in order to improve the damping of an oscillatory mode?
2. Which generators will be more effective in impacting a certain oscillatory mode?
3. What is the optimum amount of generation re-dispatch needed to improve the damping of that mode?
4. How to avoid negative interaction among multiple poorly damped modes as a result of the re-dispatch?

The answer to these questions lies in the fundamental quantity that governs the relationship between oscillatory modes and generator power output change – the eigenvalue sensitivity. The eigenvalue sensitivity quantifies the change in eigenvalue corresponding to an oscillatory mode when a particular generator output is changed by a small amount. It is not possible to write the analytical expression for this quantity in an explicit form, and so the computation of the eigenvalue sensitivity involves a perturbation based approach. That is, the generator power is changed by a small amount and the resulting change in eigenvalue is computed by calculating the new system state matrix at the new operating point.

Since the contingency screening algorithm relies on eigenvalue sensitivities, and generator outage is part of the list of contingencies to be analyzed, the idea is to perform the sensitivity computation only once at the base case, and exploit it for both the contingency screening and damping control algorithms.

6.2 Eigenvalue sensitivity to generator power output change

The first goal in arriving at the re-dispatch based damping control algorithm is to determine an accurate way to compute eigenvalue sensitivity to generator power change. There are two potential approaches to do this:

1. Approach 1: For a given generator, perturb its power output by a small amount and let the slack generator absorb the resulting change.
2. Approach 2: For a given generator, perturb its power output by a small amount and distribute the perturbation equally among the rest.

The approach that yields uniform sensitivity for a given generator, irrespective of which bus is considered as a slack bus would be considered as the suitable approach. To evaluate these 2 approaches, consider the 4-machine 2-area test system. Table 13: Sensitivity of real-part of eigenvalue to generator power using approach 1Table 13 summarizes the results for approach 1. In this table, each row represents the sensitivity of real part of eigenvalue corresponding to mode 1 for a small change in generator 1's output power as the slack bus is varied from generator 1 to generator 4.

Table 13: Sensitivity of real-part of eigenvalue to generator power using approach 1

Generator\Slack	G1	G2	G3	G4
G1	0	0.0036	0.0008	0.0042
G2	-0.0038	0	-0.0028	0.0007
G3	-0.002	0.002	0	0.0034
G4	-0.0056	-0.0014	-0.0037	0

Table 14 shows the results for approach 2. In this table, again the same convention of rows and columns is used. As mentioned earlier, the difference is in how the perturbation is distributed. In approach 2, the perturbation made at one generator is distributed equally among the rest of the generators.

Table 14: Sensitivity of real-part of eigenvalue to generator power using approach 2

Generator\Slack	G1	G2	G3	G4
G1	0.0037	0.0034	0.003	0.0028
G2	-0.0013	-0.0014	-0.0018	-0.0018
G3	0.0011	0.0013	0.0018	0.0017
G4	-0.0036	-0.0032	-0.0031	-0.0028

By comparing Table 13 and Table 14, it can be observed that in approach 2, the choice of slack bus has no impact on the eigenvalue sensitivity calculations. On the other hand, in approach 1, it can be seen that the choice of slack does have an impact on eigenvalue sensitivities. Thus, we can conclude that approach 2 is the appropriate one.

Also, from Table 14 it can be seen that for each column, the addition of sensitivities equals 0. This can be explained as follows. Let λ represent the eigenvalue corresponding to the mode of concern and N represent the total number of generators. Then, according to approach 2, a change in generation ΔP at G1, is compensated equally among the remaining generators. So, the change in eigenvalue can be approximated using Taylor series as

$$\Delta\lambda_1 = \frac{\partial\lambda}{\partial P_1}\Delta P - \frac{\partial\lambda}{\partial P_2}\left(\frac{\Delta P}{N-1}\right) - \cdots - \frac{\partial\lambda}{\partial P_N}\left(\frac{\Delta P}{N-1}\right)$$

The eigenvalue sensitivity is then given by

$$\mathrm{S}_1 = \frac{\Delta\lambda_1}{\Delta P} = \frac{\partial\lambda}{\partial P_1} - \frac{\partial\lambda}{\partial P_2}\left(\frac{1}{N-1}\right) - \cdots - \frac{\partial\lambda}{\partial P_N}\left(\frac{1}{N-1}\right)$$

Similarly, a change in generation at G2 would result in

$$\Delta\lambda_2 = -\frac{\partial\lambda}{\partial P_1}\left(\frac{\Delta P}{N-1}\right) + \frac{\partial\lambda}{\partial P_2}\Delta P - \cdots - \frac{\partial\lambda}{\partial P_N}\left(\frac{\Delta P}{N-1}\right)$$

The corresponding eigenvalue sensitivity for this change is given by

$$\mathrm{S}_2 = \frac{\Delta\lambda_2}{\Delta P} = -\frac{\partial\lambda}{\partial P_1}\left(\frac{1}{N-1}\right) + \frac{\partial\lambda}{\partial P_2} - \cdots - \frac{\partial\lambda}{\partial P_N}\left(\frac{1}{N-1}\right)$$

Similarly, the eigenvalue sensitivity for the i th generator is given by

$$\mathrm{S}_\mathrm{i} = \frac{\Delta\lambda_i}{\Delta P} = -\frac{\partial\lambda}{\partial P_1}\left(\frac{1}{N-1}\right) - \frac{\partial\lambda}{\partial P_2}\left(\frac{1}{N-1}\right) - \cdots + \frac{\partial\lambda}{\partial P_i} - \cdots \frac{\partial\lambda}{\partial P_N}\left(\frac{1}{N-1}\right)$$

Then, adding all the sensitivities for generators 1 to N gives,

$$\sum_{i=1}^{N} S_i = \frac{\partial\lambda}{\partial P_1}\left(1 - \frac{N-1}{N-1}\right) + \frac{\partial\lambda}{\partial P_2}\left(1 - \frac{N-1}{N-1}\right) + \cdots + \frac{\partial\lambda}{\partial P_N}\left(1 - \frac{N-1}{N-1}\right) = 0$$

Hence, for a given slack bus choice, the sum of all eigenvalue sensitivities to generator power changes equals 0.

6.3 Importance of eigenvalue sensitivity

Next, we demonstrate the value of using an eigenvalue sensitivity based approach for generator re-dispatch based damping control. We consider the 16-machine system for the demonstration. Each of the 16 generators is perturbed by a small amount, and the resulting change in real-part of eigenvalue corresponding to the top 3 critical modes is evaluated. Table 15 shows the resulting sensitivities.

Table 15: Sensitivity of real-part of eigenvalue to generator power change

Generator	Mode 1	Mode 2	Mode 3
G1	-0.00070	0.00018	-0.00047
G2	0.00964	0.00189	0.00618
G3	0.01350	0.00247	0.00856
G4	0.03024	0.00446	0.01128
G5	0.03882	0.00545	0.01540
G6	0.02858	0.00457	0.01487
G7	0.02483	0.00409	0.01331
G8	0.00129	0.00072	0.00059
G9	0.00194	0.00097	0.00179
G10	-0.01799	-0.00289	-0.00909
G11	-0.01828	-0.00328	-0.01007
G12	-0.01757	-0.00407	-0.01226
G13	-0.01778	-0.00405	-0.01278
G14	-0.02780	-0.00258	-0.00845
G15	-0.02560	-0.00277	-0.00920
G16	-0.02296	-0.00510	-0.00964

From Table 15 it can be seen that across all the generators, the sensitivity of real part of the eigenvalue for mode 2 is much less than that for modes 1 and 3. Also, it can be seen that mode 1 has the highest magnitude of sensitivity to generator power output changes, which indicates that controlling that mode needs much less re-dispatch as compared to the other modes. Furthermore, the signs of the sensitivities indicate the positive interaction among all the 3 critical modes. In other words, by targeting to improve mode 1 via generation re-dispatch, there would not only be an improvement in damping of mode 1, but also that of modes 2 and 3 even if they are not explicitly considered in the objective function. The eigenvalue sensitivity can be used as a first indicator to operators to determine the feasibility of generator re-dispatch based control for improvement of modal damping. As was seen in this example with mode 2, the eigenvalue sensitivity may be so low that improving the damping of such modes would require unrealistically large amount of generation re-dispatch; in which case the operators may be able to conclude that generation re-dispatch would not be a feasible solution. In other cases, such as mode 1 in the above example, the operators may use this information to further solve an optimization problem to determine re-dispatch commands for individual generating units in order to improve modal damping, as will be demonstrated in the following sub-section.

6.4 Quadratic programming based approach to decide generation re-dispatch

Next, we consider the objective of improving the damping of just mode 1 in the above 16-machine example system using generation re-dispatch. The optimization problem is formulated as a quadratic program (QP) follows:

$$\min J = \sum_{i=1}^{Ng} \Delta P_i^2$$

$$\text{Subject to: } \sum_{i=1}^{Ng} \Delta P_i = 0$$

$$\sigma_{target} = \sigma_{base} + \sum_{i=1}^{Ng} \Delta P_i S_i$$

where, Ng is the total number of generators, $\Delta \mathrm{P_i}$ is the amount of re-dispatch for generator i, $\mathrm{S_i}$ is the sensitivity of real-part of the eigenvalue to change in power of ith generator, σ_{base} is the real-part of eigenvalue for the base case, σ_{target} is the targeted real-part of eigenvalue post-dispatch.

The sensitivities are calculated using approach 2 described above. The base case real-part of eigenvalue is -0.0802. The targeted real-part is -0.13 (corresponding to a settling time of 30 seconds). Then, the QP results in a total re-dispatch of 1.0087 p.u. (i.e., the total increment within one group of generators has to be 1.0087 p.u., and an equal amount of decrement within the other group). This re-dispatch is then implemented, and the resulting post-re-dispatch eigenvalue real-part is -0.1226. As expected, the post-re-dispatch eigenvalue real-part does not exactly match with the targeted value of -0.13. This is due to the fact that the relationship between eigenvalue movement and re-dispatch of generators is expected to be non-linear one (but one for which a closed form expression is not available), but in the QP, we make the assumption of a linear relationship between generation re-dispatch and change in eigenvalue real-part given by the eigenvalue sensitivity. In spite of this assumption, it is encouraging to see that the post-dispatch eigenvalue real-part is quite close to the targeted value. We also examine the impact of this re-dispatch on mode 2, which was not explicitly considered in the objective function. However, our expectation is that because the eigenvalue sensitivities for modes 1 and 2 have the same signs, the re-dispatch would help improve the damping of mode 2 as well. The base case real-part of eigenvalue corresponding to mode 2 is -0.1242. Post-re-dispatch, this eigenvalue is -0.1342. So, although not a very significant change in magnitude, the same re-dispatch is able to move mode 2 as well in a direction that would improve its damping.

Next, we consider targeting improving the damping of just mode 2. Based on the eigenvalue sensitivities given in table 3, we expect that the improvement of damping of mode 2 would need significantly higher amounts of re-dispatch as compared to what is needed for mode 1. The base case real-part of eigenvalue corresponding to mode 2 is -0.1242. The targeted real-part is -0.1742, so that the change in real-part targeted (-0.05) matches with the previous case of mode 1. The QP

results in a total re-dispatch of 6.49 p.u. The resulting real-part of eigenvalue after the re-dispatch is -0.1427. This re-dispatch shows that for mode 2 to meet a targeted eigenvalue movement of the same magnitude as mode 1, it would require more than 6 times the re-dispatch. This finding validates our hypothesis that is based on purely base case eigenvalue sensitivities.

6.5 Damping control with constraints on amount of re-dispatch

Next, we consider the objective of improving the damping of just mode 1, with the addition of constraints on the amount of re-dispatch that each generator can provide. So, the QP formulation now has the following inequality constraints:

$$P_{min}^{i} \leq \Delta P_i \leq P_{max}^{i}$$

Note that these are constraints on the change in power of generators, and not the actual power outputs itself. For the same test case as above, we consider the following constraints: for $\mathrm{i} = 14 \text{ to } 16, \mathrm{P_{min}^{i}} = -0.15$ p.u. For the remaining generators, $\mathrm{P_{min}^{i}} = -1$ p.u. The upper limit $\mathrm{P_{max}^{i}}$ for all the generators is set at 1 p.u. The resulting change in dispatch after solving the modified QP is 1.1393 p.u. The real-part of eigenvalue post-re-dispatch is -0.1225. As expected, the addition of constraints on generator re-dispatches results in higher amount of re-dispatch (1.1393 p.u. with constraints vs. 1.0087 p.u. without constraints) to reach the same amount of modal damping.

6.6 Damping control with constraints on frequency of eigenvalue and amount of re-dispatch

Next, we consider the impact of re-dispatch on imaginary part of eigenvalue. We consider the same 16-machine system for the demonstration. Each of the 16 generators is perturbed by a small amount, and the resulting change in imaginary-part of eigenvalue corresponding to the top 3 critical modes is evaluated. Table 16 shows the resulting sensitivities.

Table 16: Sensitivity of imaginary-part of eigenvalue to generator power change

Generator	Mode 1	Mode 2	Mode 3
G1	0.00591	0.00512	0.00424
G2	-0.00217	0.00417	-0.01214
G3	-0.00505	0.00377	-0.01753
G4	-0.02360	0.00145	-0.04184
G5	-0.02635	0.00104	-0.05031
G6	-0.01989	0.00194	-0.03996
G7	-0.02042	0.00207	-0.03657
G8	0.00193	0.00428	-0.00140
G9	-0.00563	0.00402	-0.00563
G10	0.02413	0.00746	0.03171
G11	0.02482	0.00714	0.03599
G12	0.02544	0.00776	0.04753
G13	0.02546	0.00744	0.04579
G14	-0.00169	-0.02752	0.01839

G15	-0.00223	-0.01316	0.01428
G16	-0.00073	-0.01711	0.00737

We consider the objective of improving the damping of just mode 1, with the addition of constraints on the change in imaginary part of eigenvalue, along with the previous constraints on the amount of re-dispatch that each generator can provide. So, the QP formulation now has the following inequality constraints:

$$\omega_{min}^{i} \leq \sum_{i=1}^{Ng} \Delta P_i T_i \leq \omega_{max}^{i}$$

The limits are chosen as: $\omega_{min}^{i} = 0, \omega_{max}^{i} = 0.002.$ The resulting change in dispatch after solving the modified QP is 1.4125 p.u. The real-part of eigenvalue post-re-dispatch is -0.1236. As expected, the addition of constraints on imaginary part of eigenvalue in addition to constraints on generator re-dispatches results in higher amount of re-dispatch (1.4125 p.u. with additional constraints vs. 1.0087 p.u. without constraints) to reach the same amount of modal damping. The imaginary part of eigenvalue post-re-dispatch is 2.2824 (base case imaginary part is 2.2858). The post-re-dispatch imaginary part of eigenvalue is 0.0034 smaller than the base case value, although in the QP the lower limit on change in imaginary part was set at 0. This difference is due to that fact that the relationship between eigenvalue sensitivity and generator re-dispatch is a non-linear one, while in the QP formulation we are making a linearity assumption. However, for the previous case where there was no constrain on imaginary part of eigenvalue, the post-re-dispatch eigenvalue has an imaginary part of 2.3135, which is higher than the base case by 0.028. So, comparing the change in imaginary part for the 2 cases (0.0034 vs. 0.028), it is clear that the re-dispatch is able to keep a tighter control on the imaginary part. The trade-off, however, is that the amount of re-dispatch is higher than the previous case (1.4125 p.u. vs. 1.1393 p.u.).

Figure 31 below shows the percentage re-dispatch (change in generation) for each of the 16 generators for this problem with constraints on imaginary part as well as power output. As can be seen from the figure, generators 1 to 13 except 9 are required to reduce their power output, and generators 9 and 14 to 16 are required to increase their power output.

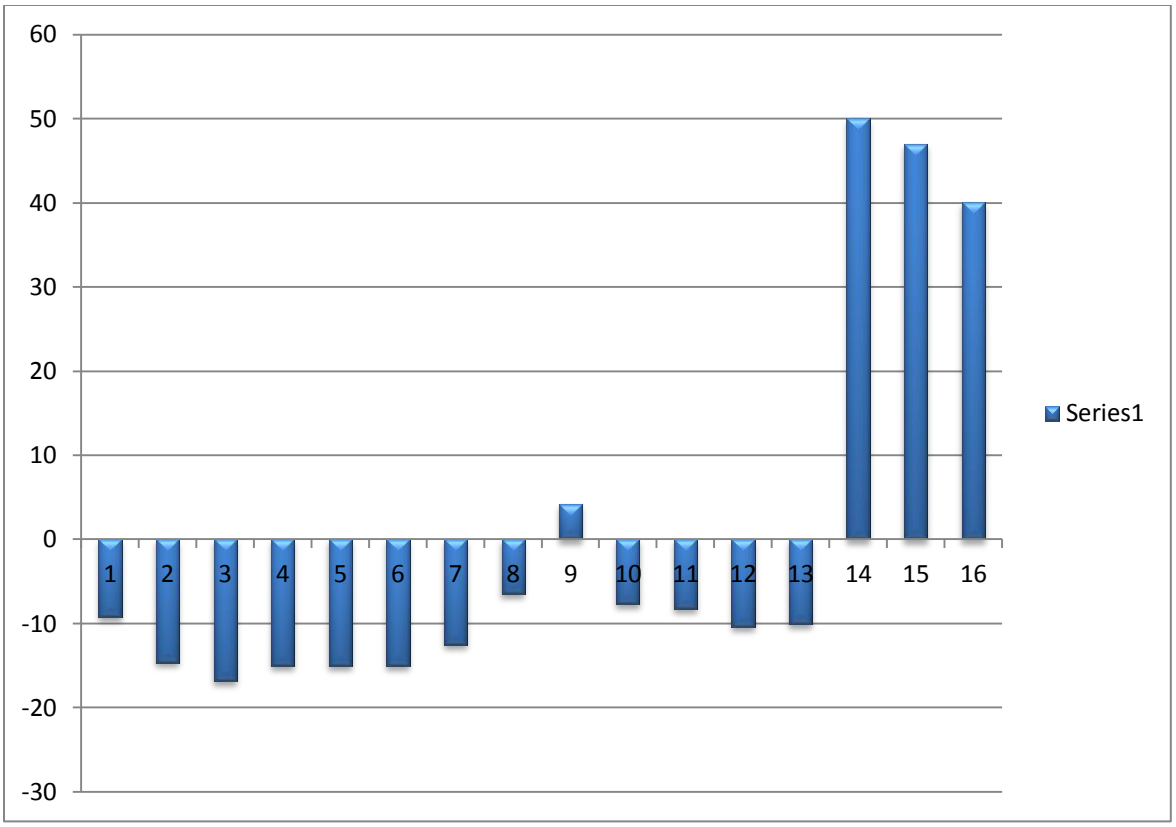


Figure 31: Individual generator and load power changes.

6.7 Proposed method of re-dispatch for ensuring post-contingency stability

In the previous sub-section, the re-dispatch command for generators was derived by posing it as an optimization problem. Constraints on the change in power dispatch and deviation of post-dispatch frequency were imposed and the optimization was solved using QP. It can be recalled that a major motivation behind model-based approach is doing 'what if' analysis – that cannot be done using measurement based approaches. To that end, the key question would be: how do we determine the generation re-dispatch in the base case scenario that could ensure stability under the post contingency conditions. The underlying challenges include:

- Considering changes in eigenvalue sensitivities following outages
- Ensuring stability across all (N-1) contingencies
- Non-linear relationship between the eigenvalue movement and re-dispatch – especially for post contingency scenario

Figure 32 shows the proposed sequence of operations in the Fast Contingency Screening and Control Action Engine (FSCAE). The fast contingency screening and ranking will be done on the base case (steps a and b) following the approach mentioned in the previous section. Based on the outcome of the screening and ranking (step b) and pre-determined threshold for the settling time, we will obtain

a list of 'critical' contingencies. If there exist no 'critical' contingencies, no control action will be considered. Otherwise, control action engine (CAE) will come into effect.

The CAE based on the eigenvalue sensitivity with respect to generator power output for the Base Case was described in the previous sub-sections. However, the eigenvalue sensitivity under post-contingency condition is likely to change. To address this challenge, we propose the following:

Step I: Compute the sensitivities $\frac{\Delta\sigma}{\Delta P}$ and $\frac{\Delta\omega}{\Delta P}$ for each generator under post-worst case contingency condition. The sensitivity is computed only for the mode of interest which is causing instability.

Step II: Under current operating condition (base case), use the above computed sensitivity for optimization. The targeted real part of the eigenvalue will be determined in two steps. First, the targeted change in the real part is determined by the equation

$$\Delta\sigma_{\text{target}} = \sigma_{\text{post-target}} - \sigma_{\text{post}}$$

Here, σ_{post} is the real-part of the eigenvalue under the worst-case post contingency condition (which is known based on the post contingency model) and $\sigma_{\text{post-target}}$ is the targeted real-part of the eigenvalue under the same condition (which can be set by the operator). Next, the target of the base case eigenvalue real-part under post-dispatch condition is determined using the value of $\Delta\sigma_{\text{target}}$ as $\sigma_{\text{target}} = \Delta\sigma_{\text{target}} + \sigma_{\text{base}}$.

Step III: Evaluate the dispatch command by the quadratic optimization mentioned before, to achieve $\boldsymbol{\sigma_{\text{target}}}$ with constraints on real power and frequency. This leads to a new base case with a target settling time as shown in step (e) of Figure 32.

Step IV: To validate its effectiveness, time-domain simulations will be run on the most 'critical' contingencies using the new operating condition following the re-dispatch (step f). Using Prony analysis the settling time will be calculated. If it is acceptable, the re-dispatch command will be initiated. Otherwise, the CAE will have to be run again.

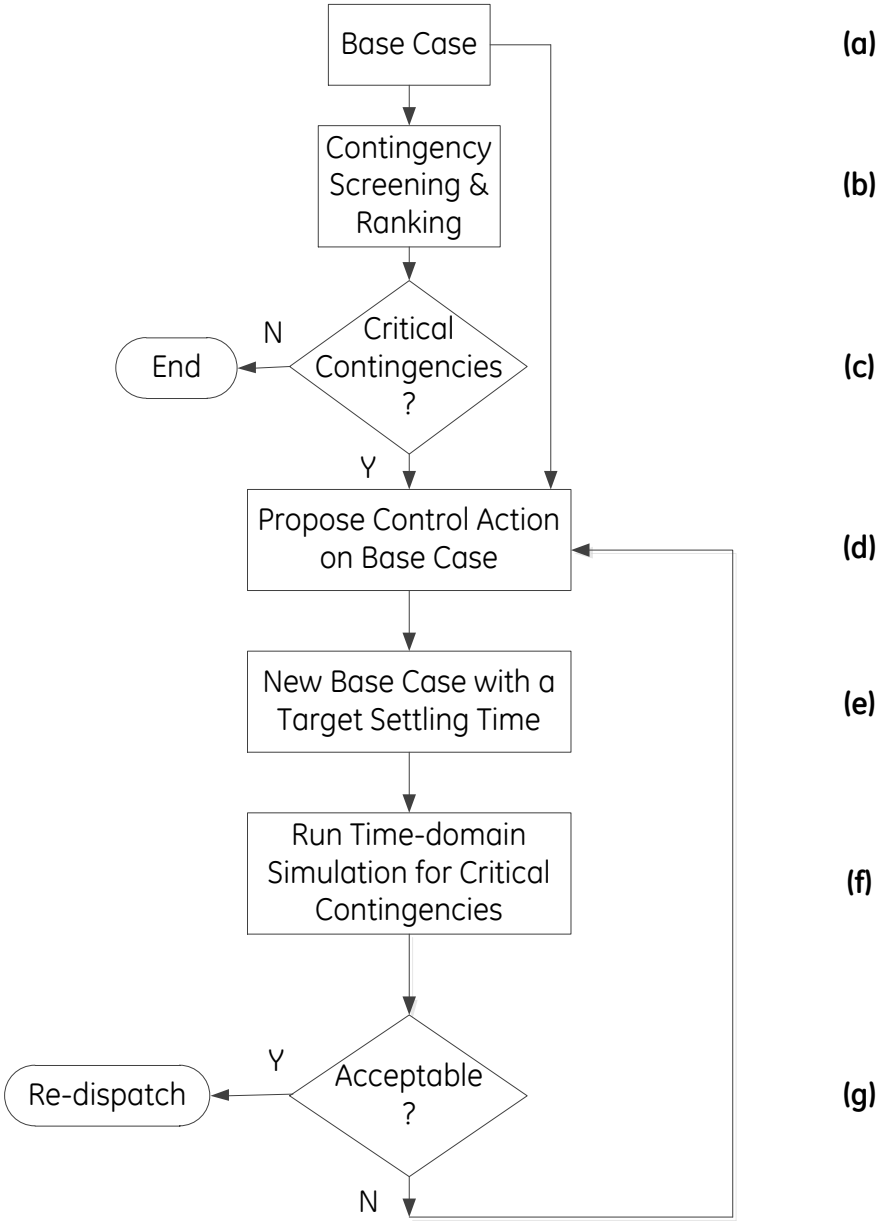


Figure 32: Proposed sequence of operations in the Fast Contingency Screening and Control Action Engine (FSCAE).

6.8 Results

In the 16-machine 68 bus test system, the line outage scenarios lead to instability of mode 1. From contingency screening results the worst case contingency is line outage #46 (bus 37 – bus 68) whose post contingency real part of the eigenvalue is 0.234, which is in the far right half of the s-plane. It should be noted that instability of such severity under (N-1) contingency condition is not practical for a realistic power system.

To circumvent this impractical contingency, the next worst contingency, i.e. #23 (bus 22 – bus 21) where the real part of the mode 1 eigenvalue is 0.123 is chosen. Let us consider that we would like to attain a 30s settling time for this mode under the post-contingency scenario. This leads to $\sigma_{post-target} = -0.13$. Therefore, $\Delta\sigma_{target} = -0.13 - 0.123 = -0.253$ and σ_{target} =-0.253 – 0.08 = -0.333.

Table 17 shows the eigenvalue sensitivities computed for the outage of the line #23. These sensitivities were used under the base case and the QP was used to derive the power dispatch to achieve σ_{target} = -0.333 with constraints imposed on change in dispatch and modal frequency. The limits on the change in dispatch were set to be 20% of their nominal values. The lower and upper limits on modal frequency were the same as the previous case study.

The real part of the eigenvalue under the pre and post-dispatch (Case I) condition are shown in Table 18. The results from Case I shows that the base case σ is moved to -0.21 instead of -0.33. For outage #23, it moves to -0.097 as opposed to the target of -0.13. It can also be seen that stability is achieved for top 10 critical contingencies.

As a test, the line impedance of line #23 was increased to a very high value to represent the outage. As an example (Case II), the eigenvalue sensitivities were computed with the corresponding line impedance increased by 800% as shown in Table 17. When these sensitivities were used for re-dispatch, the real part of the eigenvalue under post-dispatch condition when line #23 is out becomes -0.135 as compared to the target of -0.13 (see Table 18). It can also be observed that post-contingency conditions are stable.

Table 17: Eigenvalue sensitivities computed for contingency #23

Generator	Line out (#23)		Increased impedance by 800%	
	$\frac{\Delta\sigma}{\Delta P}$	$\frac{\Delta\omega}{\Delta P}$	$\frac{\Delta\sigma}{\Delta P}$	$\frac{\Delta\omega}{\Delta P}$
G1	-0.0097	0.0109	-0.0037	0.0074
G2	-0.0016	0.0039	0.0056	0.0007
G3	0.0025	0.0008	0.0096	-0.002
G4	0.0283	-0.0191	0.0316	-0.0196
G5	0.0341	-0.0196	0.0388	-0.0208
G6	0.0927	-0.0505	0.052	-0.0275
G7	0.0845	-0.0553	0.046	-0.0301
G8	-0.0075	0.008	-0.0015	0.0043
G9	-0.0067	0.0018	-0.0006	-0.0026
G10	-0.0253	0.025	-0.0209	0.0225
G11	-0.0259	0.0255	-0.0214	0.023
G12	-0.0261	0.0268	-0.0213	0.0239
G13	-0.0266	0.0274	-0.0217	0.0242
G14	-0.0395	0.0054	-0.0331	-0.0009
G15	-0.0377	0.0042	-0.031	-0.0017
G16	-0.0351	0.0044	-0.0281	-0.0008

Table 18: Real part of mode 1 Eigen value under pre-dispatch and post-dispatch conditions for base case and top 10 contingencies

	Pre-dispatch	Post-dispatch (Case I)	Post-dispatch (Case II)
Outage #	σ	σ	σ
Outage #46 (bus 37 – bus 68)	0.234	-0.0339	-0.1106
Outage #23 **(bus 22 – bus 21)**	**0.123**	**-0.0966**	**-0.1348**
Outage #22 (bus 21 - bus 68)	0.025	-0.1537	-0.1892
Outage #80 (bus 61- bus 60)	0.008	-0.1499	-0.1947
Outage #81 (bus 61 – bus 60)	0.008	-0.1499	-0.1947
Outage #29 (bus 27 – bus 37)	-0.005	-0.1666	-0.2007
Outage #63 (bus 47 – bus 53)	-0.031	-0.1348	-0.1577
Base Case	**-0.08**	**-0.2108**	**-0.2366**

Figure 33 and Figure 34 graphically shows the result shown in Table 18. It is apparent that extent of the movement of σ from pre-dispatch to post-dispatch condition shows consistency across different contingencies.

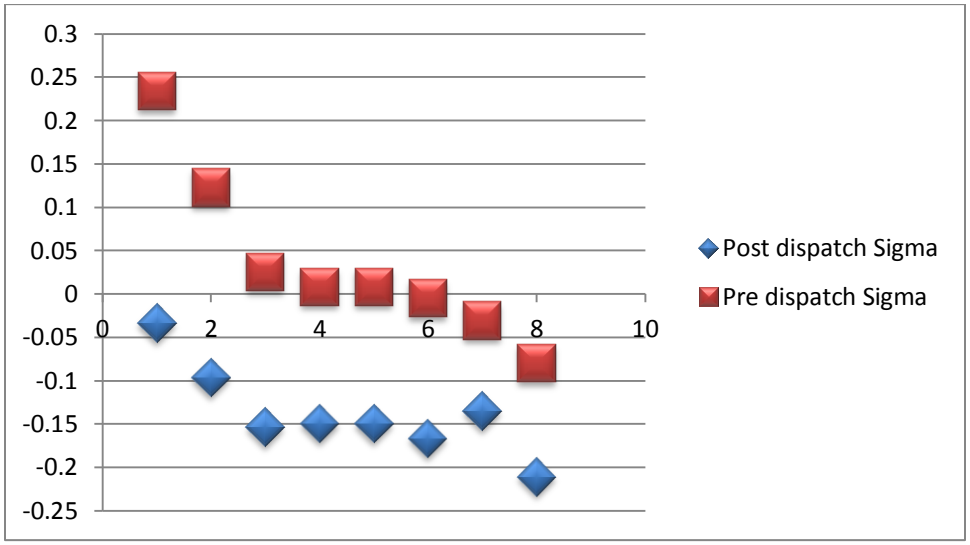


Figure 33: Case I: Real part of Eigen value (Y-axis) for Pre and post-dispatch conditions. X-axis plots top 10 contingencies and the base case.

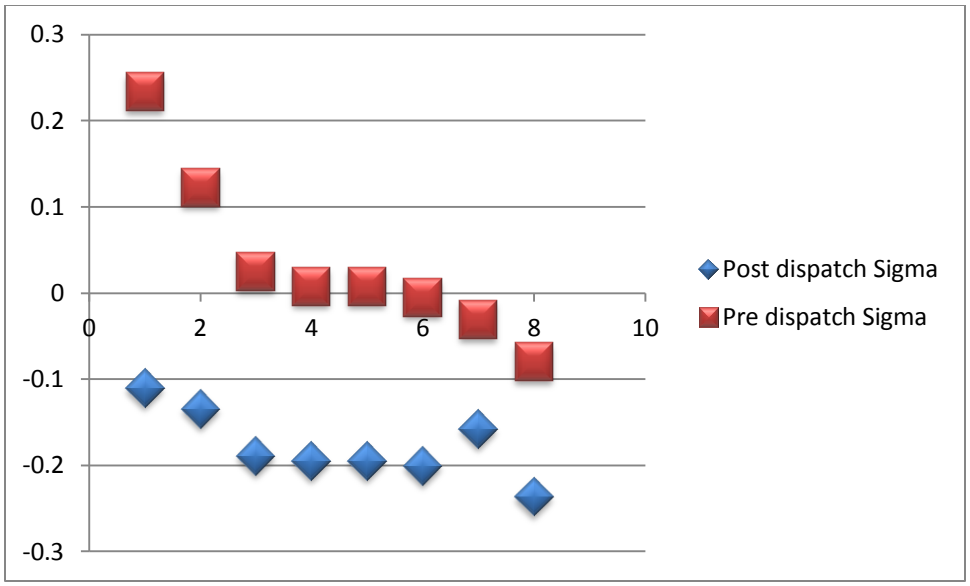


Figure 34: Case II: Real part of Eigen value (Y-axis) for Pre and post-dispatch conditions. X-axis plots top 10 contingencies and the base case.

6.9 Eigenvalue sensitivity with loads

So far, we considered the objective of improving damping of oscillatory modes using generation re-dispatch. Next, we consider the possibility of load participation in this objective. That is, if both generation and load are allowed to change, how would the algorithm be designed to take advantage of this added flexibility. The approach is very similar to the previous one. The key addition is in the way eigenvalue sensitivity to load change is calculated.

There are a couple of differences when considering load flexibility vs. generation change. When considering load as an actuator, one constraint is that loads can be changed in only one direction. That is, it is possible to only reduce the load, and not increase it, unlike the case with generation re-dispatch. Secondly, there are many more load points in the system compared to the generators. Typically, if a system has tens of generators, there would be several hundreds to thousands of aggregate loads. So, it would be infeasible to compute the eigenvalue sensitivity for each of the loads.

The approach proposed here allows one to determine eigenvalue sensitivity for loads in a realistic manner. The steps of the eigenvalue sensitivity computation with respect to loads are as follows:

Step I: Compute the generating areas using previously computed eigenvalue sensitivity information for generators – group 1 corresponds to positive sensitivities, and group 2 to negative

Step II: Decrease aggregated loads L1 in group 1 and distribute the decrement among all generators to compute $S_L = \frac{\Delta\lambda}{\Delta P_L}$

Step III: Repeat Step II to compute eigenvalue sensitivity for group 2 loads L2

Table 19 below shows the sensitivity of real part of eigenvalue for load change for the 16 machine system using the approach described above.

Table 19: Sensitivity of real-part of eigenvalue to load change

Load group	Mode 1	Mode 2	Mode 3
L1	-0.00469	-0.00064	-0.00382
L2	0.01939	0.00337	0.00942

Table 20 below shows the sensitivity of imaginary part of eigenvalue for load change.

Table 20: Sensitivity of imaginary-part of eigenvalue to load change

Load group	Mode 1	Mode 2	Mode 3
L1	-0.00540	-0.00497	-0.00161
L2	-0.01541	0.00060	-0.02639

From Table 19 and Table 20 it can be seen that a decrease in group 2 loads increases damping (moves eigenvalue to left) of all the critical modes. So, this aids generation reduction for generators with positive sensitivity. This information can be incorporated in the damping control algorithm as follows:

The QP problem can be modified to

$$\min J = \sum_{i=1}^{Ng} \Delta P_i^2 + \sum_{i=1}^{2} \Delta P_{Lgrp}^2$$

$$\text{Subject to: } \sum_{i=1}^{Ng} \Delta P_i + \sum_{i=1}^{2} \Delta P_{L_{grp}} = 0$$

$$\sigma_{target} = \sigma_{base} + \sum_{i=1}^{Ng} \Delta P_i S_i + \sum_{i=1}^{2} \Delta P_{L_{grp}} S_{L_{grp}}$$

$$P_{min}^{i} \leq \Delta P_i \leq P_{max}^{i}$$

$$P_{min}^{L} \leq \Delta P_{L_{grp}} \leq 0$$

The modified QP was tested on the 16 machine test system. The resulting total reduction in generation is found to be 0.9569pu (this is aided by a total load reduction of 0.15 pu). The total increase in generation is 1.1069 pu (compared to the previous case which was 1.1393 pu). The resulting post-dispatch eigenvalue real-part is -0.1289 (compared to the previous case which was -0.1225). Therefore, it can be seen that including loads in the problem improves post-dispatch

eigenvalue while reducing the total generation re-dispatch compared to the case when considering just generation re-dispatch. Figure 35 below shows the percentage re-dispatch (change in power) for each of the 16 generators for this problem with constraints on imaginary part as well as power output as well as for the group of loads. The bars labeled 17 and 18 represent the load changes (positive and negative respectively).

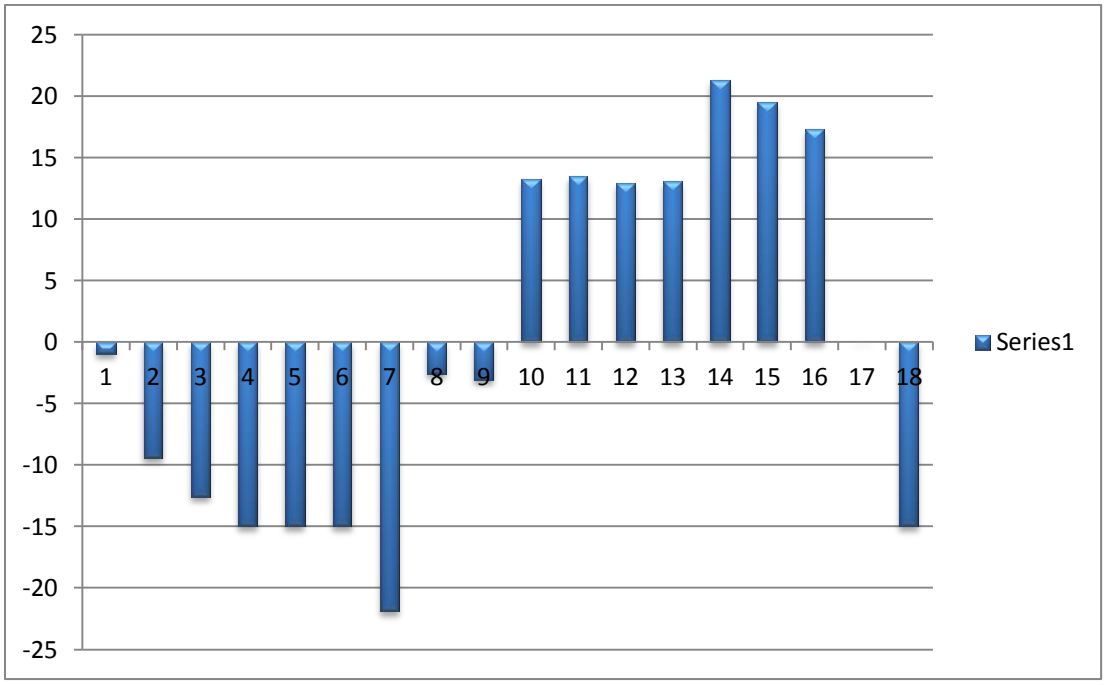


Figure 35: Individual generator and load power changes considering post-worst contingency.

6.10 Grid operational benefit from fast screening and control

The fast contingency screening and control action engine (FSCAE) will provide significant benefit to the system operation by improving the grid resiliency to fast time-scale phenomena including:

- Improved utilization of existing transmission assets by reduction of stability margin.
- Increased turbine life by reducing frequency oscillation in the systems
- Avoid false tripping of backup relays during power swings by maintaining more than 5% damping ration.
- Reduction of the dynamic rating of expensive FACTS devices used for supplementary damping control action.
- Improved situational awareness with real-time dynamic simulation capability.

7 References

[1] J. Hauer, D. J. Trudnowski, G. Rogers, B. Mittlestadt, W. Litzenberger and J. Johnson, "Keeping an eye on power system dynamics," *IEEE Comput. Appl. Power,* vol. 10, no. 4, pp. 1-10, 1997.

[2] J. Hauer, D. Trudnowski and J. G. DeSteese, "A perspective of WAMS analysis tools for tracking of oscillatory dynamics," in *IEEE Power Engineering Society General Meeting*, 2007.

[3] G. Liu and V. Venkatasubramanian, "Oscillation monitoring from ambient PMU measurements by frequency domain decomposition," in *IEEE International Symposium of Circuits and Systems (ISCAS 2008)*, 2008.

[4] D. J. Trudnowski, J. W. Pierre, N. Zhou, J. F. Hauer and M. Parashar, "Performance of three mode-meter block-processing algorithms for automated dynamic security assessment," *IEEE Transactions on Power System,* vol. 23, no. 2, pp. 680-690, 2008.

[5] N. Zhou, D. J. Trudnowski, J. W. Pierre and W. A. Mittelstadt, "Electromechanical mode online estimation using regularized robust RLS methods," *IEEE Transactions on Power System,* vol. 23, no. 4, pp. 1670-1680, 2008.

[6] "NASPI," [Online]. Available: https://www.naspi.org/.

[7] J. F. Hauer, C. J. Demeure and L. L. Scharf, "Initial results in prony analysis of power system response signals," *IEEE Transactions on Power System,* vol. 5, no. 1, pp. 80-89, 1990.

[8] I. Kamwa, R. Grondin, E. J. Dickinson and S. Fortin, "A minimal realization approach to reduced-order modelling and modal analysis for power system response signals," *IEEE Transactions on Power System,* vol. 8, no. 3, pp. 1020-1029, 1993.

[9] J. H. Sanchez-Gasca and J. H. Chow, "Performance comparison of three identification methods for the analysis of electromechanical oscillations," *IEEE Transactions of Power System,* vol. 14, no. 3, pp. 995-1002, 1999.

[10] "ABB PSGuard828," [Online]. Available: http://www05.abb.com/global/scot/scot221.nsf/veritydisplay/344dc0a1adfd774ec12570d0002ed127/$file/998_psguard828_datasheet.pdf.

[11] "SIEMENS SIGUARD Solutions," [Online]. Available: http://www.energy.siemens.com/us/pool/hq/services/power-transmission-distribution/power-technologies-international/network-consulting/FT06%20System%20security%20%28SIGUARD%20DSA%29%20111202.pdf.

[12] "AREVA e-terravision," [Online]. Available: http://www.rpmbrasil.com.br/smartgrid2008/sergio_yoshio_fujii.pdf.

[13] "FERC/NERC staff report on Sept 8, 2011 Arizona - Southern California Blackout," [Online]. Available: http://www.ferc.gov/legal/staff-reports/04-27-2012-ferc-nerc-report.pdf.

[14] T. Davis, Direct methods for sparse linear systems, 1st edition, SIAM, 2006.

[15] Y. Saad, Iterative methods for sparse linear systems, 3rd edition, SIAM, 2000.

[16] A. Gupta and V. Kumar, "Parallel algorithms for forward and back substitution in direct solution of sparse linear systems," in *ACM/IEEE conference on supercomputing*, 1995.

[17] F. Pruvost and T. Cadeau, "Numerical accelerations for power systems transient stability simulations," in *17th power systems computation conference (PSCC)*, Stockholm, 2011.

[18] T. Davis, "Algorithm 832: UMFPACK v4.3 - an unsymmetric-pattern multifrontal method," *ACM transactions on mathematical software,* 2004.

[19] T. Davis and E. Natarajan, "Algorithm 907: KLU, a direct sparse solver for circuit simulation problems," *ACM transactions on mathematical software,* 2010.

[20] B. Stott, "Power system dynamic response calculations," *Proceedings of the IEEE,* vol. 67, no. 2, pp. 219-241, 1979.

[21] F. L. Alvarado, "Parallel solution of transient problems by trapezoidal integration," *IEEE Trans. Power App. Syst.,* Vols. PAS-98, no. 3, pp. 1080-1090, 1979.

[22] M. Benzi, "Preconditioning techniques for large linear systems: a survey," *Journal of Comput. Physics,* vol. 182, no. 2, pp. 418-477, 2002.

[23] S. K. Khaitan, J. D. McCalley and Q. Chen, "Multifrontal solver for online power system time-domain simulation," *IEEE Transactions on Power Systems,* vol. 23, no. 4, pp. 1727-1737, 2008.

[24] P. M. Anderson and A. A. Fouad, Power System Control and Stability, New York: IEEE Press, 2003.

[25] Y. Dong and H. R. Pota, "Fast transient stability assessment using large step-size numerical integration," *IET Gener. Trans. Dis.,* vol. 138, no. 4, pp. 377-383, 1991.

[26] P. W. Sauer and M. A. Pai, "Power system steady-state stability and the load-flow Jacobian," *IEEE Transactions on Power Systems,* vol. 5, no. 4, pp. 1374-1383, 1990.

[27] A. Bose, "Application of direct methods to transient stability analysis of power systems," *IEEE Transactions on Power Apparatus and Systems,* Vols. PAS-103, no. 7, pp. 1629-1636, 1984.

[28] F. A. Rahimi, M. G. Lauby, J. N. Wrubel and K. L. Lee, "Evaluation of the transient energy function method for on-line dynamic security analysis," *IEEE Transactions of Power Systems,* vol. 8, no. 2, pp. 497-507, 1993.

[29] T. Jianzhong, C. Hsiao-Dong and T. P. Conneen, "A sensitivity-based BCU method for fast derivation of stability limits in electric power systems," *IEEE Transactions on Power Systems,* vol. 8, no. 4, pp. 1418-1428, 1993.

[30] M. J. Laufenberg and M. A. Pai, "Sensitivity theory in power systems: Application in dynamic security analysis," in *IEEE International Conference on Control Applications*, 1996.

[31] Y. Xue, T. V. Custem and M. Ribbens-Pavella, "Real-time analytic sensitivity method for transient security assessment and preventive control," in *IEE Generation, Transmission and Distribution*, 1988.

[32] V. brandwajn, A. B. R. Kumar, A. Ipakchi, A. Bose and S. D. Kuo, "Severity indices for contingency screening in dynamic security assessment," *IEEE Transactions on Power Systems,* vol. 12, no. 3, pp. 1136-1142, 1997.

[33] W. Li and A. Bose, "A coherency based rescheduling method for dynamic security," in *20th International Conference on Power Industry Computer Applications*, 1997.

[34] C. Fu and A. Bose, "Contingency ranking based on severity indices in dynamic security analysis," *IEEE Transactions on Power Systems,* vol. 14, no. 3, pp. 980-985, 1999.

[35] H. K. Nam, K. S. Shim, Y. K. Kim, S. G. Song and K. Y. Lee, "Contingency ranking for transient stability via eigen-sensitivity analysis of small signal stability model," in *IEEE Power Engineering Society Winter Meeting*, 2000.

[36] P. W. Sauer, "On the formulation of power distribution factors for linear load flow methods," *IEEE Transactions on Power Apparatus and Systems,* Vols. PAS-100, no. 2, pp. 764-770, 1981.

[37] M. Liu and G. Gross, "Framework for the design and analysis of congestion revenue rights," *IEEE Transactions on Power Systems,* vol. 19, no. 1, pp. 243-251, 2004.

[38] R. H. Yeu, "Small signal analysis of power systems: Eigenvalue tracking method and eigenvalue estimation contingency screening for DSA," PhD Dissertation; University of Illinois at Urbana-Champaign, 2010.

[39] M. A. Pai, Energy Function Analysis for Power System Stability, Boston, MA: Kluwer, 1989.

[40] X. Liu, "Power system dynamic vulnerability under extreme transmission line contingencies," Master's Thesis; McGill University, 2007.

[41] P. Kundur, Power System Stability and Control, New York: McGraw-Hill, 1994.

[42] B. Pal and B. Chaudhuri, Robust Control in Power Systems, ser. Power Electronics and Power Systems, New York: Springer, 2005.

[43] G. Rogers, Power System Oscillations, Boston: Kluwer Academic, 2000.

8 Publications

The following is the list of resulting publications and patent applications from this program:

1. C. A. Baone, N. Acharya, S. Veda and N. R. Chaudhuri, "Fast contingency screening and ranking for small signal stability assessment," in *Proceedings of the IEEE PES General Meeting*, National Harbor, MD, July 2014.

2. C. A. Baone, N. R. Chaudhuri and N. Acharya, "System and method for analyzing oscillatory stability in electrical power transmission systems," Patent filed, October 2013.

3. C. A. Baone, N. R. Chaudhuri and N. Acharya, "Systems and methods for improved generation re-dispatch of power generation systems," Patent *under preparation*.
"

4. Zhenyu Huang, Ruisheng Diao, Di Wu, Shuangshuang Jin, Yu Zhang, Yousu Chen, Bin Zheng, "A time-stacking method for dynamic simulations" Patent *filed, November 2014*.

5. Zhenyu Huang, Ruisheng Diao, Di Wu, Shuangshuang Jin, Yu Zhang, Yousu Chen, Bin Zheng, Time Stacking Method for Power System Dynamic Simulation" *to be submitted to an IEEE or similar conference.*

Appendix A - Model data and system

The dynamic models of two test systems (4 machine 11 bus system and 16 machine 68 bus system) that are widely used in stability studies are developed in PSLF environment. These test systems will be used in the project to test the fundamental concepts for contingency screening and control action for small signal stability.

A.1 A 4 Machine 11 Bus Test System

Figure 36 shows a 4 machine 11 bus test system that is used to study small signal oscillation problem [41]. This system consists of two similar areas connected by weak tie-lines. Each area consists of two coupled units each having a rating of 900 MVA at 20 kV. A step-up transformer with impedance of 15% on 900 MVA and 20/230 kV base, and an off-nominal tap ratio of 1.0 connects each generator to the transmission system. The transmission system nominal voltage is 230 kV. The system is operating with area 1 (consisting of buses 1, 2, 5, 6 and 7) exporting 400 MW to area 2 (consisting of buses 3, 4, 9, 10 and 11). The active component of the loads has constant current and reactive component has constant impedance characteristics. Two shunt capacitors with ratings of 200 MVAR and 350 MVAR are connected at buses 7 and 9 respectively. All four generators are represented by a 6th order sub-transient model and are equipped with IEEE DC1A type exciters.

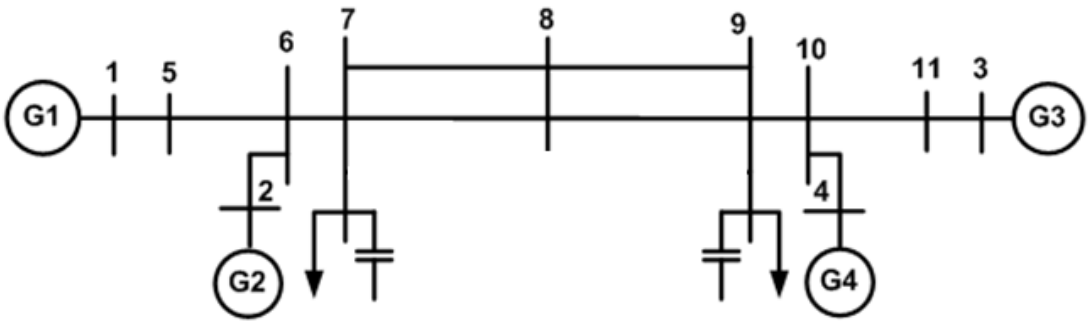


Figure 36: 4 machine 11 bus test system.

A.2 A 16 Machine 68 Bus Test System

Figure 37 shows a 16-machine, 68 bus test system. Further details can also be found in [42]. This is a reduced order equivalent of the interconnected New England test system (NETS) and New York power system (NYPS) which is taken from [43]. The buses are renumbered as in [42] keeping the topology and the data (static and dynamic) the same as in [43]. There are five geographical regions out of which NETS and NYPS are represented by a group of generators whereas, import from each of the three other neighboring areas, Area #3, Area #4 and Area #5 are approximated by equivalent generator models.

Generators G1 to G9 are the equivalent representation of the NETS generation whilst, G10 to G13 represent the generation of the NYPS. Generators G14 to G16 are the dynamic equivalents of the three neighboring areas connected to the NYPS. There are three major transmission corridors

between NETS and NYPS connecting buses #60-#61, #53-#54 and #27-#53. All these corridors have double-circuit tie-lines.

All generators are represented by their 6th order sub-transient model. The generators G1 to G8 are equipped with slow excitation systems (IEEE-DC1A) whilst G9 is equipped with a fast acting static excitation system (IEEE ST1A) and a speed-input power system stabilizer (PSS) to ensure adequate damping for its local models. The rest of the generators are under manual excitation control.

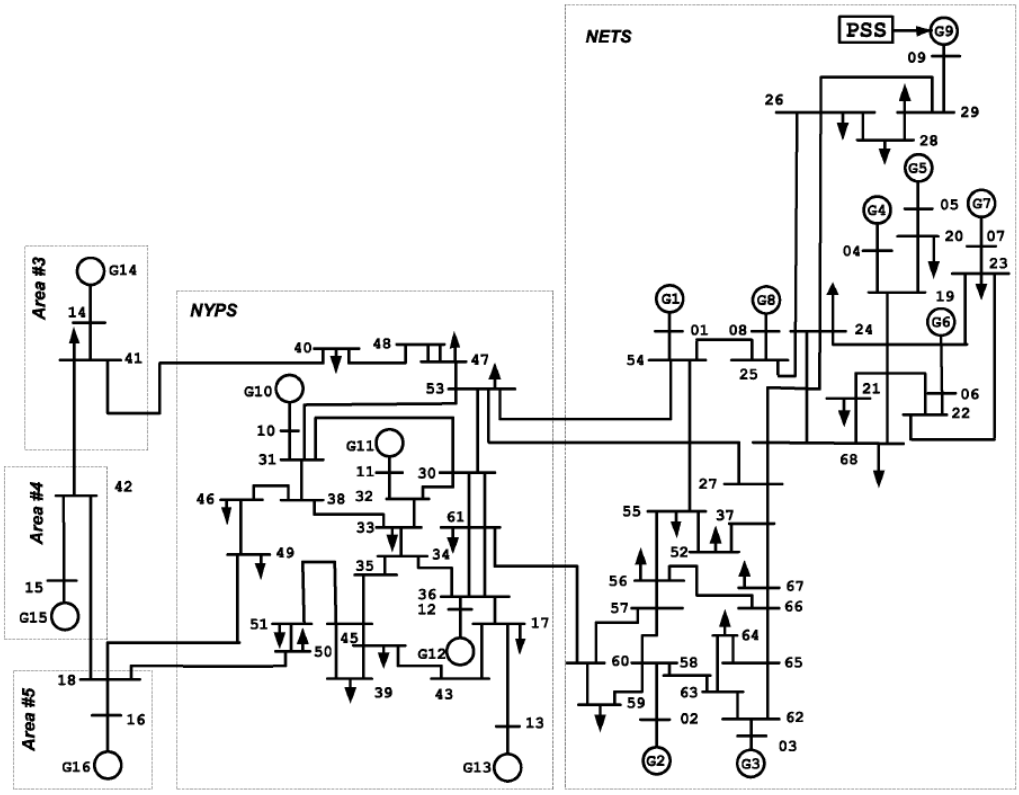


Figure 37: 16 machine 68 bus dynamic equivalent of New York – New England power system.

Appendix B - Benchmark model

B.1 Introduction

The 4-machine 11-bus system (shown in Figure 36 and the 16-machine 68-bus system (shown in Figure 37) have been accepted as standard test cases that are widely used by power systems engineers in the industry as well as the academia to study the small signal stability phenomenon. The results of these two test cases have been published in numerous peer-reviewed publications. These test cases are available with the project team as a MATLAB/SIMULINK model. The fact that MATLAB can provide us a linearized model of the test network gives us an analytic foundation for testing our hypotheses towards developing techniques for contingency screening, ranking and control action engine. The results of a linear analysis of the MATLAB/SIMULINK model will be matched with the published results, thus giving us a certain degree of confidence in the accuracy of the model. Since the high performance computing (HPC) component of the project will be based on time domain simulation in PSLF, it is necessary that we have the test case available in PSLF also. In order to ensure the accuracy of the PSLF model and its time domain simulation, the team will benchmark the results of the PSLF model against the benchmarked MATLAB model, for both systems.

B.2 Objective

The main objective is to benchmark the two systems in PSLF against the same model created in MATLAB/SIMULINK by comparing their load flow and dynamic results. While it is ideal to have the two models to exactly replicate each other, the innate characteristics of the software like numerical integration techniques, mathematical model definition of the network components, etc. would introduce some differences in the model behavior. It is important to quantify these differences irrespective of how subtle they are, in order to understand and estimate the differences between the final outcomes. Although the 4-machine 11-bus system and 16-machine 68-bus system is available as a standardized test case for studying small signal stability, there might be subtle differences in the model performance depending on the software platform on which the model was created. While these differences are inconsequential in studies that use the same software for analysis, they must be quantified in our project since we seek to leverage a combination of multiple methods and computational software to achieve our project objectives.

MATLAB is used widely by scientists and engineers across several disciplines that require mathematical modeling. Simulink is a block diagram environment for modeling, simulating and analyzing dynamic systems. The 4-machine 11-bus system and 16-machine 68-bus system is modeled by the first principles using the differential-algebraic equations that govern the dynamic response of the power system components. The small signal behavior of the model has been validated against published results.

GE Concorda PSLF is a commercial power systems analysis tool that is designed to provide comprehensive and accurate load flow, dynamic simulation and short circuit analysis for large real-life electric networks. PSLF is used widely by utilities for planning and operation. The use of PSLF for small signal analysis allows the team to use larger networks for testing and validating the developed techniques which would otherwise have not been possible with MATLAB/SIMULINK alone.

B.3 Approach

In order to compare the network modeled in two different simulation environments, it is necessary to characterize the differences in model behavior during dynamic and linear analysis. The differences should be quantified and the possible sources of differences be identified and ensure the closest possible matching between the models in the two simulation environments.

B.4 Benchmarking results for 4-machnine 11-bus system

Dynamic simulation

A dynamic simulation of the network provides a time domain response of the various power system components. The dynamics of the network is modeled as a set of differential-algebraic equations that represent all the time-dependent components of the network like generators and exciters. Without any disturbance, the dynamics of the network should exhibit a flat profile, indicating steady initial conditions. Once this is achieved, the dynamic response of the two models can be compared by initiating a disturbance. The dynamic responses of the two models are compared by initiating a disturbance. Two types of disturbances, namely generator output step change and bus fault were used for the benchmarking process of the 4-machine 11-bus system.

Generator output step change is the smaller of the two disturbances. In this case, a pulse was created by increasing the output of generator 1 by 0.01 p.u. and decreasing the output of generator 3 by 0.01 p.u. Bus fault is a large disturbance. A three phase symmetrical fault (self-clearing) is placed at bus 8 for a short period of time and then cleared.

The rotor angles of generators 1, and 2 with generator 3 as a reference are plotted over the duration of simulation for each of the two models and are presented in Figure 38-Figure 43. The red plots represent the PSLF output, the blue lines are from the MATLAB simulation.

(1) Constant Current Load Model (Small Disturbance)

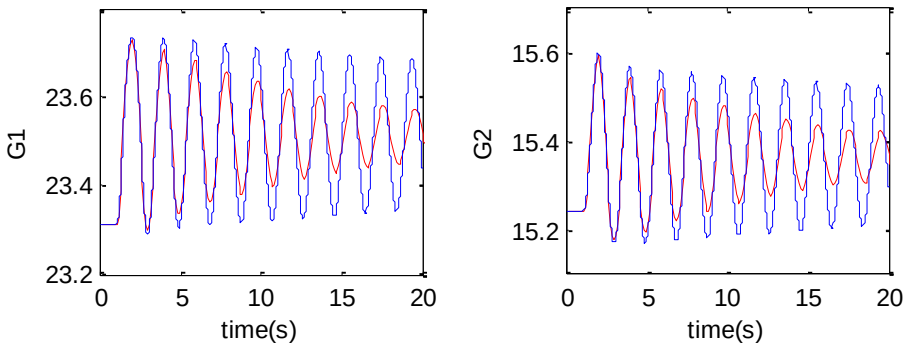


Figure 38: Rotor angle plot for a small disturbance with constant current load model.

(2) Constant Current Load Model (Large Disturbance)

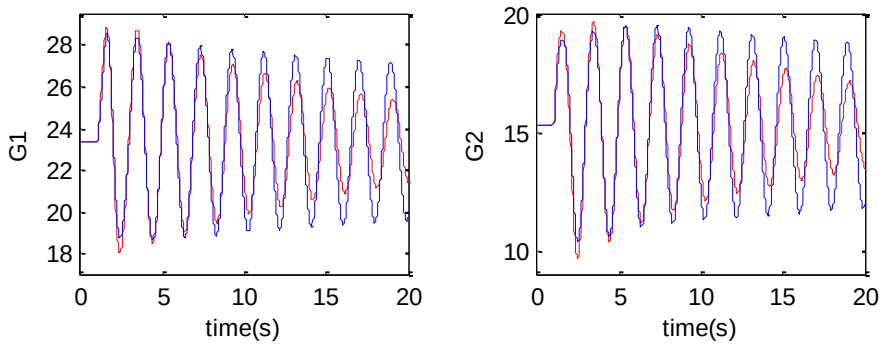


Figure 39: Rotor angle plot for a large disturbance with constant current load model.

(3) Default Load Model (Small Disturbance)

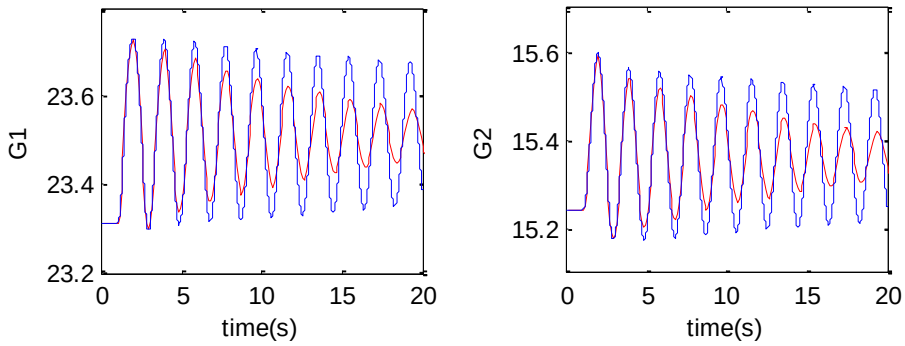


Figure 40: Rotor angle plot for a small disturbance with default load model.

(4) Default Load Model (Large Disturbance)

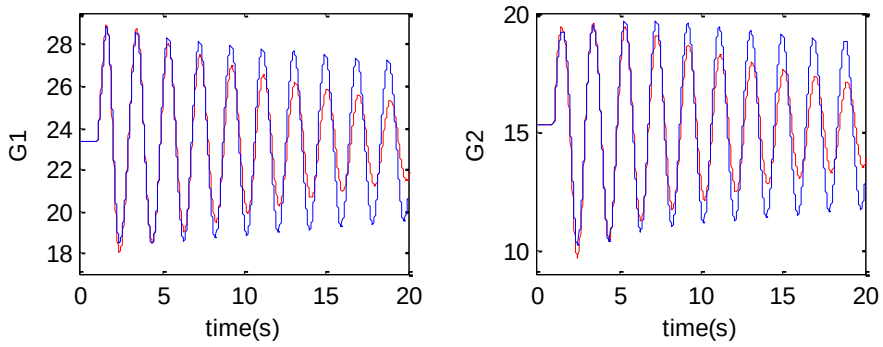


Figure 41: Rotor angle plot for a large disturbance with default load model.

(5) Constant Impedance Load Model (Small Disturbance)

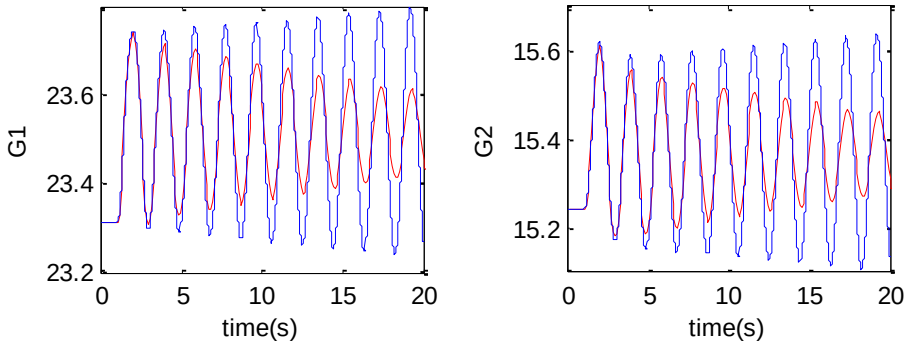


Figure 42: Rotor angle plot for a small disturbance with constant impedance load model.

(6) Constant Impedance Load Model (Large Disturbance)

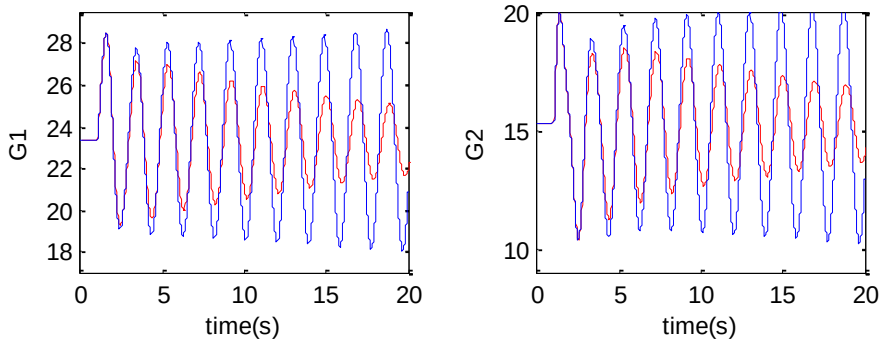


Figure 43: Rotor angle plot for a small disturbance with constant impedance load model.

The comparison of the time-domain response of the MATLAB/SIMULINK model and PSLF model shows that although the frequency of oscillation is same after the disturbance, the damping are different. The mismatch is due to the differences in representation of the dynamic components like generators and exciters in various software tools. The generator and exciter model in the MATLAB/SIMULINK is based on the model described in [41] (which forms the basis of this test system). While the commercial tools like PSLF requires that the model be in a standard modeling format accepted by the industry and standard associations. For example "genrou" for solid rotor generator represented by equal mutual inductance rotor modeling, "gentpf" for generator represented by uniform inductance ratios rotor modeling to match WSCC type F model with shaft speed effects neglected, "gensal" for salient pole generator represented by equal mutual inductance rotor modeling and so forth.

Small signal response

The 4-machine system has only one critically damped inter-area mode. This mode (shown in Table 21) is given in [41]. Default load model (50% constant current, 50% constant impedance) is used.

Table 21: Critically damped inter-area mode of the 4-machine system from [41].

Case	Critical Eigenvalues		Frequency (Hz)	Damping Ratio (%)	Settling Time (s)
	Real	Imaginary			
Manual Excitation	-0.111	3.43	0.545	3.2	36
DC Excitation	-0.018	3.27	0.52	0.5	222

The linearization results from MATLAB/SIMULINK are shown in Table 22. The results shown in Table 21 and Table 22 are close enough, which means the model built in MATLAB/SIMULINK is an accurate representation of the 4 machine system.

Table 22: Linearized results of MATLAB/SIMULINK model.

Case	Critical Eigenvalues		Frequency (Hz)	Damping Ratio (%)	Settling Time (s)
	Real	Imaginary			
Manual Excitation	-0.115	3.42	0.544	3.4	34.7
DC Excitation	-0.017	3.27	0.521	0.5	236.2

Finally, the results of prony analysis for the MATLAB/SIMULINK model and the PSLF model are compared and shown in Table 23 and

Table 24. Here are some observations that can be obtained from the comparison:

1. The frequency of the MATLAB/SIMULINK model and PSLF model are very close;

2. The damping ratio in PSLF is larger than that in MATLAB/SIMULINK for all cases. Since the generator model, static exciter models and load models used in MATLAB/SIMULINK and PSLF are slightly different; these can be the probable factors causing discrepancies in the small signal responses.

Table 23: Comparison of prony results for MATLAB/SIMULINK and PSLF models (Manual Excitation).

Manual Excitation Case		MATLAB/SIMULINK			PSLF		
		Frequency (Hz)	Damping (%)	Settling Time (s)	Frequency (Hz)	Damping (%)	Settling Time (s)
Small Disturbance (pulse)	Const. I	0.5435	3.85	30.43	0.5400	4.71	25.01
	Const. Z	0.5395	2.05	57.65	0.5453	4.13	28.30
	Default	0.5460	3.72	31.38	0.5434	4.44	26.40
Large Disturbance (bus fault)	Const. I	0.5438	3.39	34.55	0.5425	4.48	26.20
	Const. Z	0.5414	2.56	45.90	0.5416	3.90	30.17
	Default	0.5454	3.37	34.66	0.5448	4.33	26.99

Table 24: Comparison of prony results for MATLAB/SIMULINK and PSLF models (DC Excitation).

DC Excitation Case		MATLAB/SIMULINK			PSLF		
		Frequency (Hz)	Damping (%)	Settling Time (s)	Frequency (Hz)	Damping (%)	Settling Time (s)
Small Disturbance (pulse)	Const. I	0.5165	0.47	261.88	0.5129	2.41	51.42
	Const. Z	0.5216	-0.42	293.66	0.5183	1.3	94.36
	Default	0.5202	0.52	236.99	0.5159	2.24	55.02
Large Disturbance (bus fault)	Const. I	0.5162	0.46	266.30	0.5111	1.81	68.85
	Const. Z	0.5221	-0.31	387.53	0.5176	1.59	77.45
	Default	0.5152	0.44	279.07	0.5155	1.87	65.94

B.5 Benchmarking results for 16-machnine 68-bus system

Dynamic simulation

Similar benchmarking process was done for the 16-machine 68-bus system. The system is modeled in MATLAB/SIMULINK and PSLF. The differences in model behavior during dynamic simulation are compared. Three different types of load modeling (1. 50% constant impedance, 50% constant current, 2. Constant impedance, 3. Constant Current) and two different types of disturbances (1. Small disturbance by perturbing generator mechanical power input, 2. Large disturbance by applying fault at bus 53 and clearing the fault by opening line 53-54), yielding a total of six cases on which the models were compared.

The rotor angles of generators 14, 15 and 16 with generator 13 as a reference are plotted over the duration of simulation for each of the two models and are presented in Figure 44-Figure 48. The red plots represent the PSLF output, the blue lines are from the MATLAB simulation and the green lines represent the difference between the two simulation results at each sample of the simulation.

(1) Constant Current Load Model (Small Disturbance)

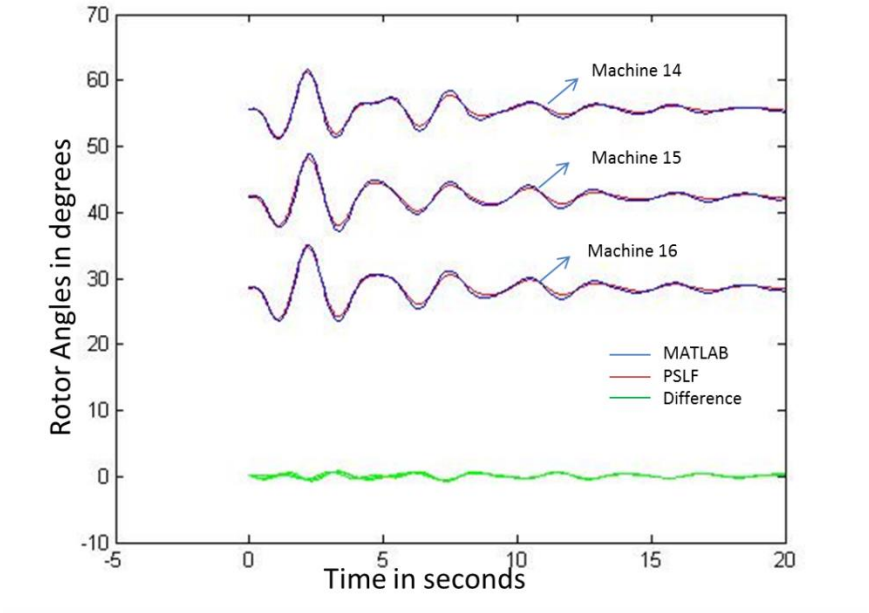


Figure 44: Rotor angle plot for a small disturbance with constant current load model.

(2) Constant Current Load Model (Large Disturbance)

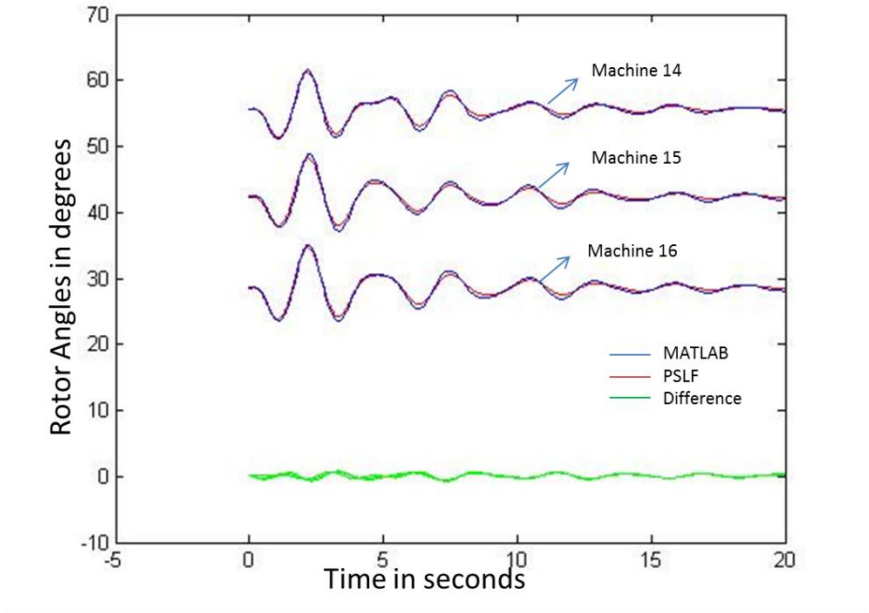


Figure 45: Rotor angle plot for a large disturbance with constant current load model.

(3) Default Load Model (Small Disturbance)

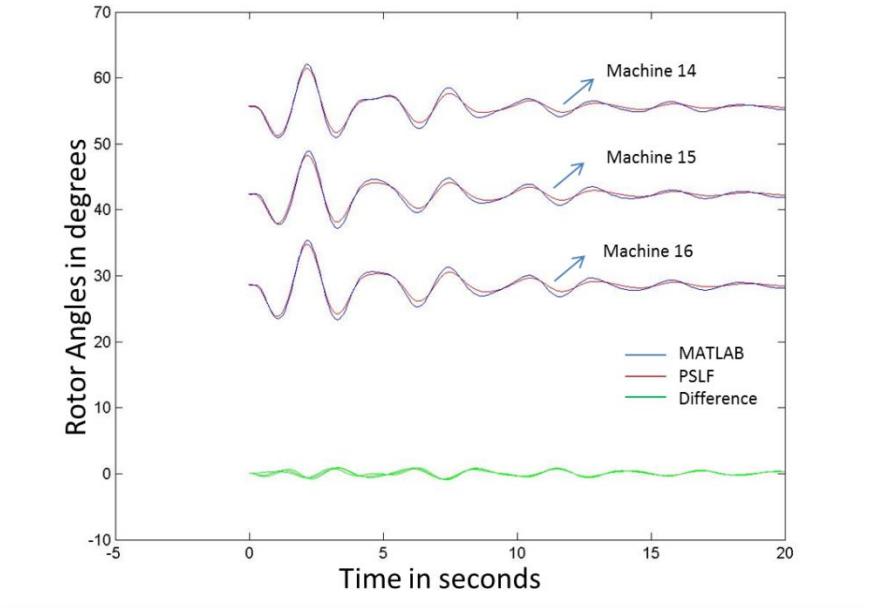


Figure 46: Rotor angle plot for a small disturbance with default load model.

(4) Default Load Model (Large Disturbance)

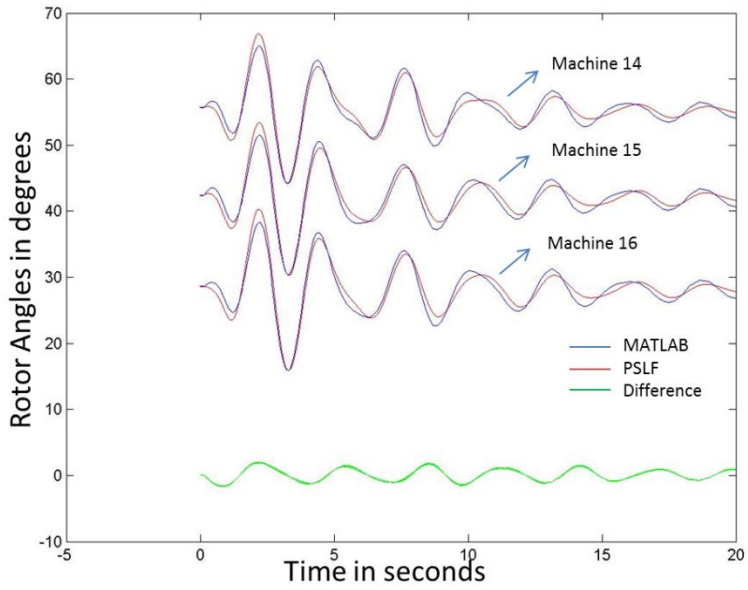


Figure 47: Rotor angle plot for a large disturbance with default load model.

(5) Constant Impedance Load Model (Small Disturbance)

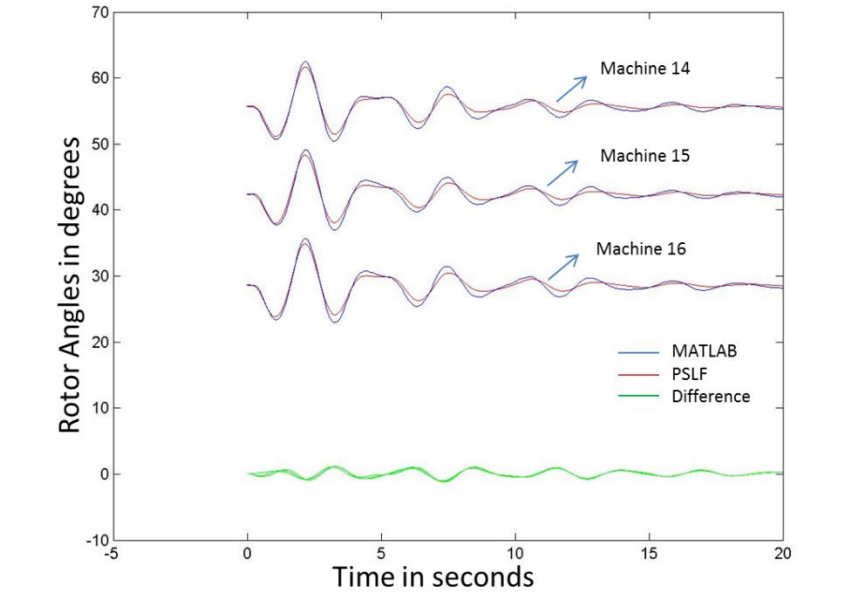


Figure 48: Rotor angle plot for a small disturbance with constant impedance load model.

(6) Constant Impedance Load Model (Large Disturbance)

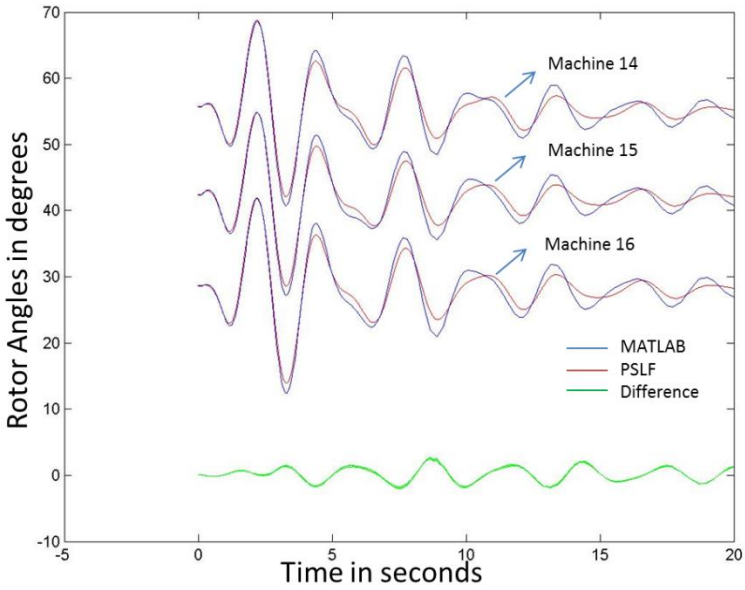


Figure 49: Rotor angle plot for a small disturbance with constant impedance load model.

Small signal response

In order to accomplish the benchmarking, it is necessary that the team perform the linearization of the Simulink model. Since the Simulink model was built using the fundamental equations, the results from this exercise present the "true modes" of the network. Table 25 and Table 26 list the four critically damped inter-area modes in the system for base case and for line 53-54 outage case.

There are four critically damped inter-area modes in the 16-machine test system, as shown Table 25 and Table 26.

Table 25: Linearized results of MATLAB/SIMULINK model for base case.

Load Model	Modes	Frequency(Hz)	Damping (%)	Settling Time (s)
Constant I	Mode 1	0.366	5.9	29.5
	Mode 2	0.505	4.4	28.7
	Mode 3	0.587	5.7	18.9
	Mode 4	0.789	5.0	16.0
Constant Z	Mode 1	0.367	6.3	27.6
	Mode 2	0.492	4.4	29.5
	Mode 3	0.582	5.2	21.1
	Mode 4	0.788	5.0	16.2
Default	Mode 1	0.368	5.9	29.2
	Mode 2	0.498	4.4	28.2
	Mode 3	0.588	5.5	19.5
	Mode 4	0.789	5.0	16.1

Table 26: Linearized results of MATLAB/SIMULINK model for line 53-54 out.

Load Model	Modes	Frequency(Hz)	Damping (%)	Settling Time (s)
Constant I	Mode 1	0.357	5.3	33.8
	Mode 2	0.503	4.3	29.1
	Mode 3	0.558	5.1	22.2
	Mode 4	0.789	5.0	16.0
Constant Z	Mode 1	0.351	5.3	34.0
	Mode 2	0.491	4.3	30.4
	Mode 3	0.556	4.5	25.5
	Mode 4	0.788	5.0	16.2
Default	Mode 1	0.355	5.4	33.2
	Mode 2	0.497	4.3	29.5
	Mode 3	0.560	5.0	22.9
	Mode 4	0.788	5.0	16.1

The small signal properties of the MATLAB/SIMULINK and PSLF response for different modes using different signals had been analyzed. The results of the prony analysis listed in Tables 3, 4, 5 and 6.

Table 27: Comparison of prony results for MATLAB/SIMULINK and PSLF models (Mode 1).

Mode 1		MATLAB/SIMULINK			PSLF		
		Frequency (Hz)	Damping (%)	Settling Time (s)	Frequency (Hz)	Damping (%)	Settling Time (s)
Small	Const. I	0.367	5.8	30.1	0.352	7.1	25.6
	Const. Z	0.367	6.2	27.9	0.348	6.5	28.2
	Default	0.368	5.9	29. 4	0.352	6.9	26.1
Large	Const. I	0.356	5.1	35.3	0.331	5.7	34.1
	Const. Z	0.351	5.2	34.7	0.333	6.0	31.7
	Default	0.356	5.3	34.1	0.332	5.4	35.6

Table 28: Comparison of prony results for MATLAB/SIMULINK and PSLF models (Mode 2).

Mode 2		MATLAB/SIMULINK			PSLF		
		Frequency (Hz)	Damping (%)	Settling Time (s)	Frequency (Hz)	Damping (%)	Settling Time (s)
Small	Const. I	0.506	4.7	27.1	0.485	4.6	28.2
	Const. Z	0.497	4.9	26.1	0.463	5.3	25.9
	Default	0.498	5.1	25.2	0.501	5.1	25.2
Large	Const. I	0.504	5.3	23.9	0.501	5.6	22.6
	Const. Z	0.503	5.5	23.2	0.515	4.6	26.4
	Default	0.499	5.2	24.4	0.499	6.2	20.6

Table 29: Comparison of prony results for MATLAB/SIMULINK and PSLF models (Mode 3).

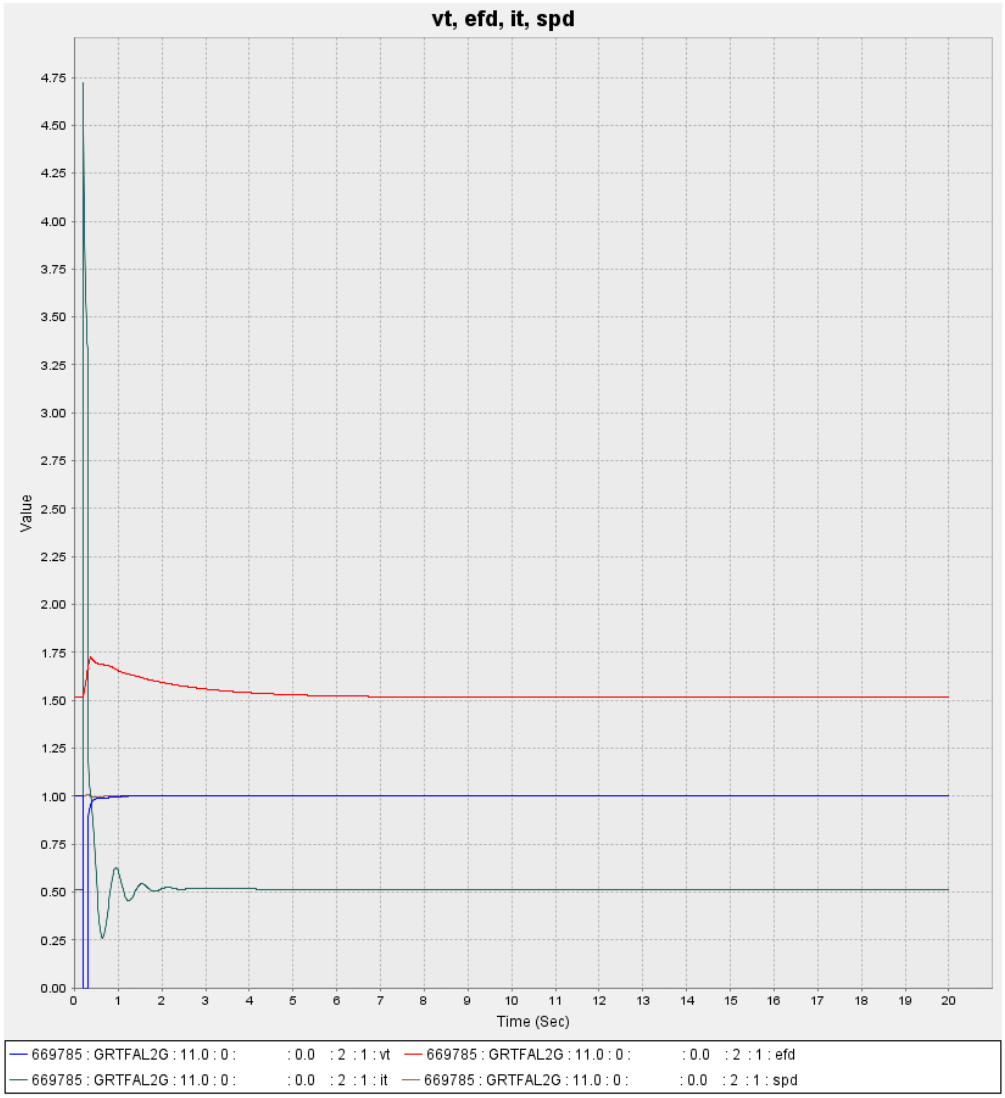
Mode 3		MATLAB/SIMULINK			PSLF		
		Frequency (Hz)	**Damping (%)**	**Settling Time (s)**	**Frequency (Hz)**	**Damping (%)**	**Settling Time (s)**
Small	Const. I	0.584	6.0	18.2	0.592	6.2	20.6
	Const. Z	0.579	5.4	20.4	0.579	5.8	19.2
	Default	0.586	5.7	19.1	0.585	7.0	15.5
Large	Const. I	0.558	4.6	25.1	0.556	5.9	19.3
	Const. Z	0.555	4.2	27.1	0.558	6.1	18.7
	Default	0.560	4.8	24.0	0.558	5.9	19.2

Table 30: Comparison of prony results for MATLAB/SIMULINK and PSLF models (Mode 4).

Mode 4		MATLAB/SIMULINK			PSLF		
		Frequency (Hz)	Damping (%)	Settling Time (s)	Frequency (Hz)	Damping (%)	Settling Time (s)
Small	Const. I	0.781	3.9	20.8	0.821	6.1	33.9
	Const. Z	0.778	4.2	19.6	0.795	4.7	17.0
	Default	0.771	4.2	19.6	0.780	3.4	23.9
Large	Const. I	0.815	5.9	13.3	0.796	4.7	17.2
	Const. Z	0.786	5.9	13.7	0.810	5.9	13.3
	Default	0.787	5.1	15.8	0.807	8.0	9.9

Comparison of Tables 2 to 5 shows that the model matches closely in terms of their small signal behavior. Some difference in the results are expected due to the differences in the dynamic model being implement in the two platform to represent the same component.

Appendix C – PSLF simulation results on laptop

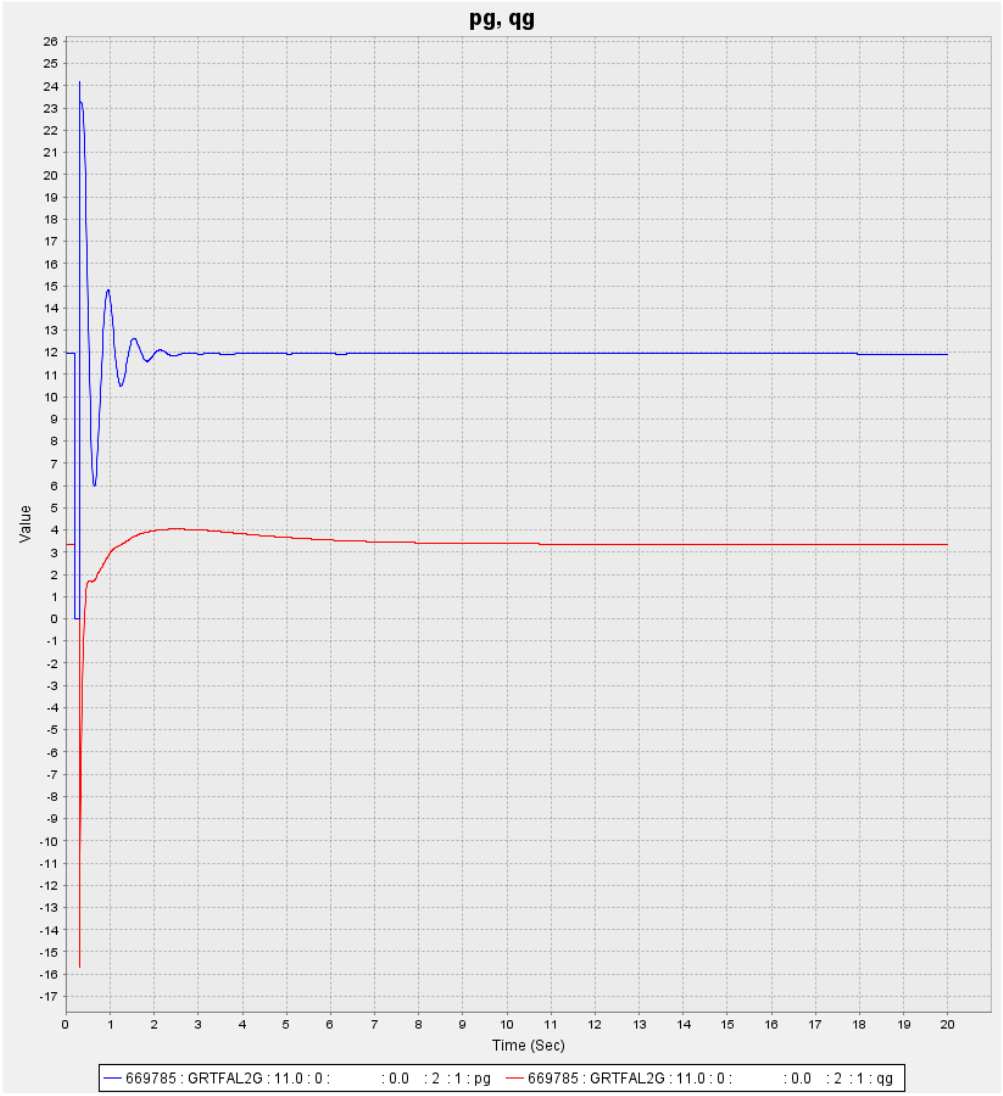



```
0,   100.00, 32, 0, 1, 60.00     / PSS®E-32.1     WED, JUL 03 2013   8:3
0,   100.00, 32, 0, 1, 60.00     / PSS®E-32.1     WED, JUL 03 2013   8:3
```

pslf.chf

Sat Dec 06 16:17:59 2014

Figure 50: Single core PSLF simulation showing original variable response

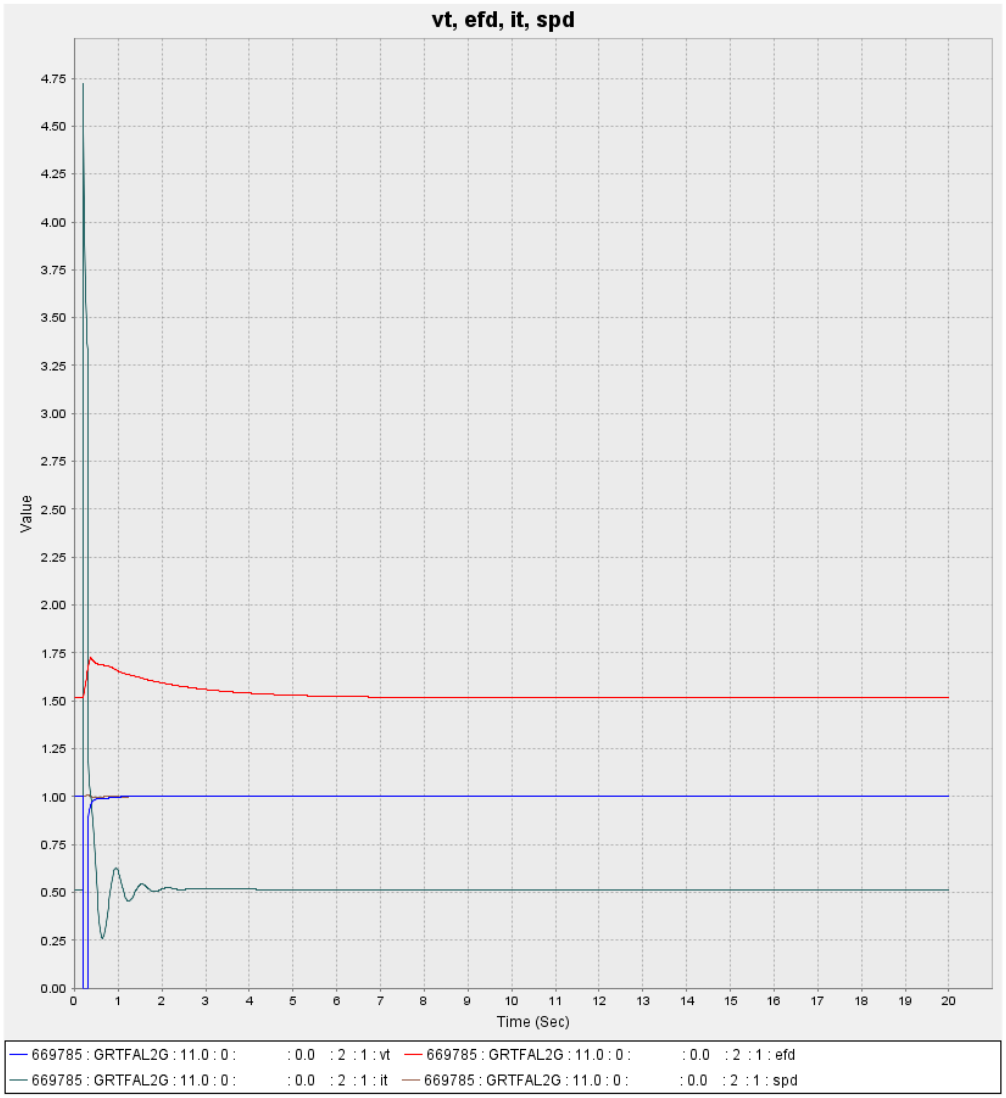



```
0,   100.00, 32, 0, 1, 60.00     / PSS®E-32.1     WED, JUL 03 2013   8:3
0,   100.00, 32, 0, 1, 60.00     / PSS®E-32.1     WED, JUL 03 2013   8:3
```

pslf.chf

Sat Dec 06 16:18:31 2014

Figure 51: Single core PSLF simulation showing original variable response



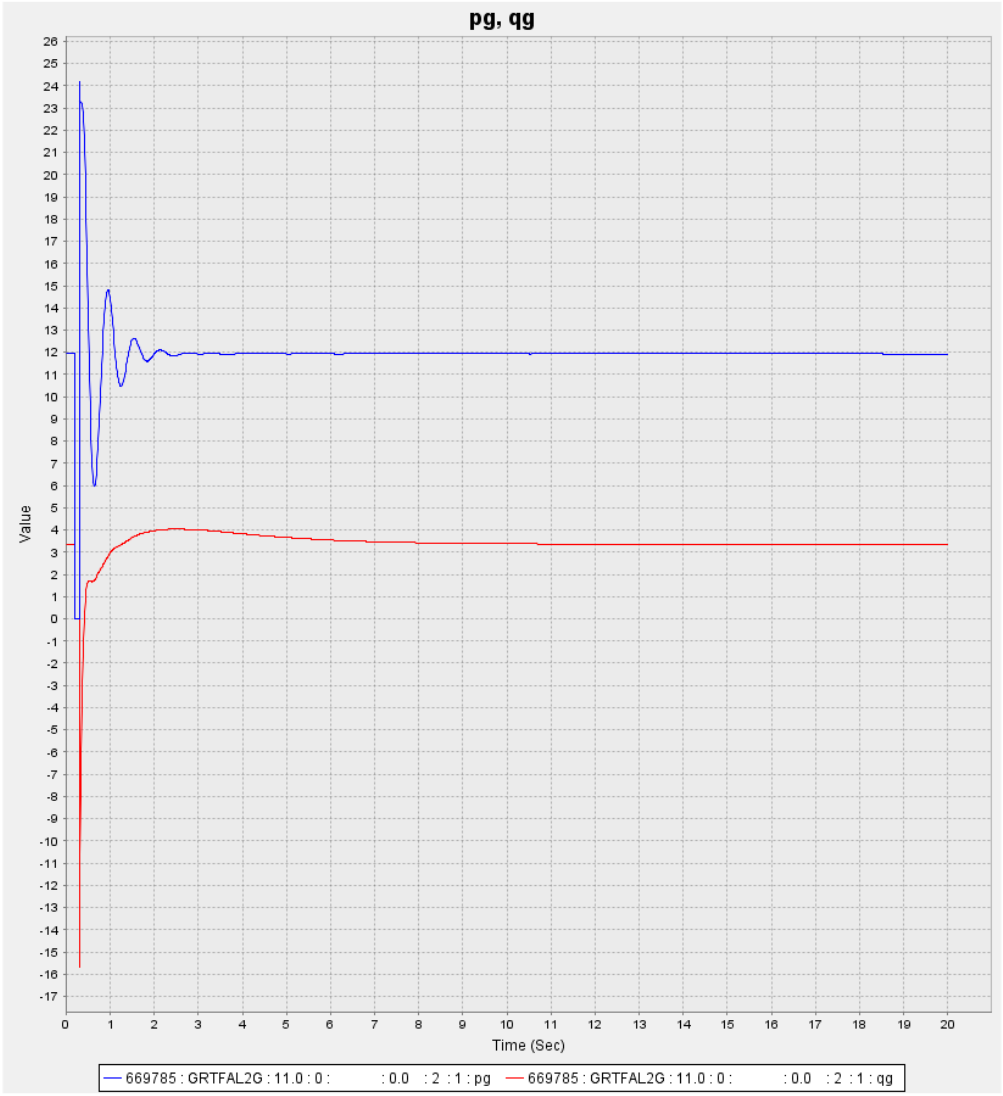

```
0,   100.00, 32, 0, 1, 60.00     / PSS®E-32.1    WED, JUL 03 2013   8:3
0,   100.00, 32, 0, 1, 60.00     / PSS®E-32.1    WED, JUL 03 2013   8:3
```

pslf.chf

L_PIC_RUNS

Sat Dec 06 17:21:37 2014

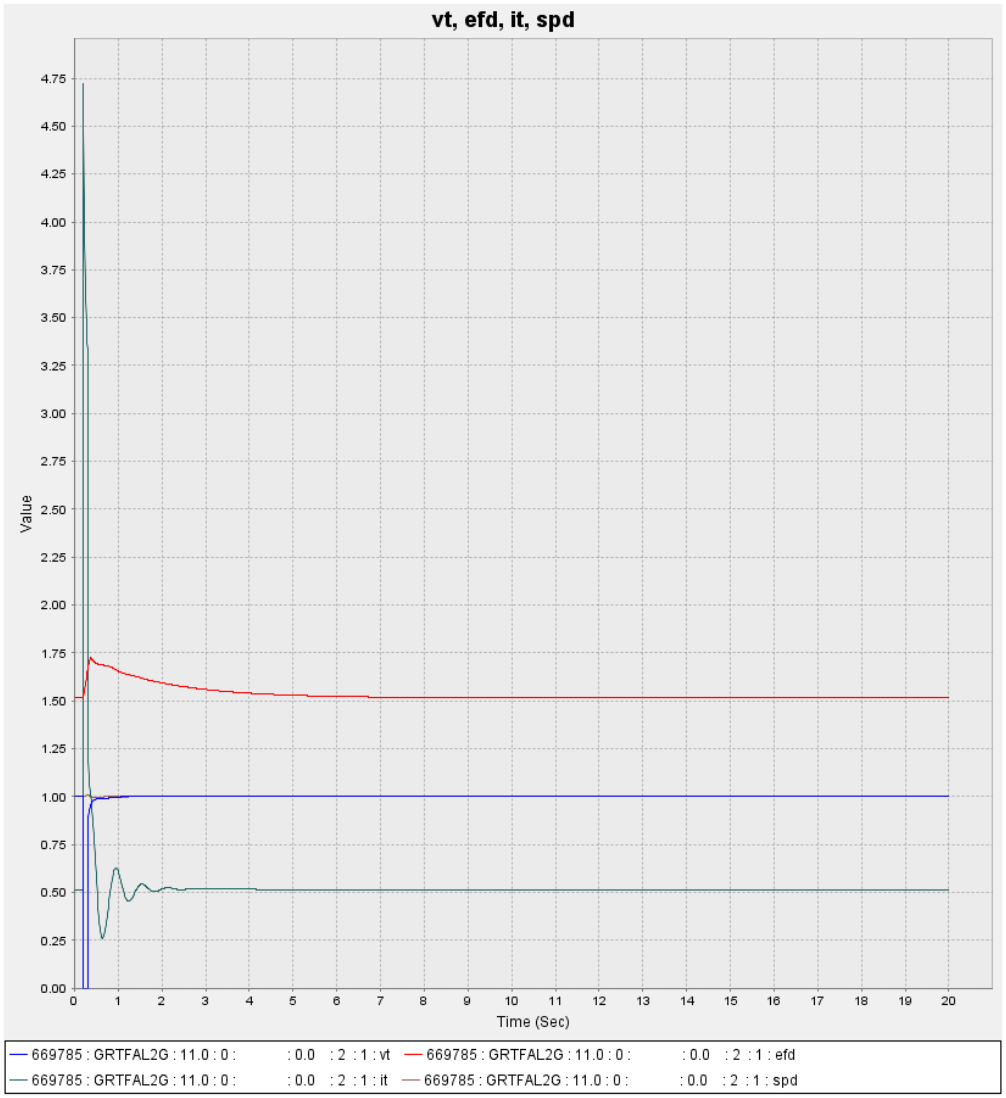
Figure 52: 2 threads + new linear solver version of PSLF showing that the simulation accuracy remains unchanged




```
0,   100.00, 32, 0, 1, 60.00     / PSS®E-32.1     WED, JUL 03 2013   8:3
0,   100.00, 32, 0, 1, 60.00     / PSS®E-32.1     WED, JUL 03 2013   8:3
```

pslf.chf L_PIC_RUNS
Sat Dec 06 17:22:01 2014

Figure 53: 2 threads + new linear solver version of PSLF showing that the simulation accuracy remains unchanged

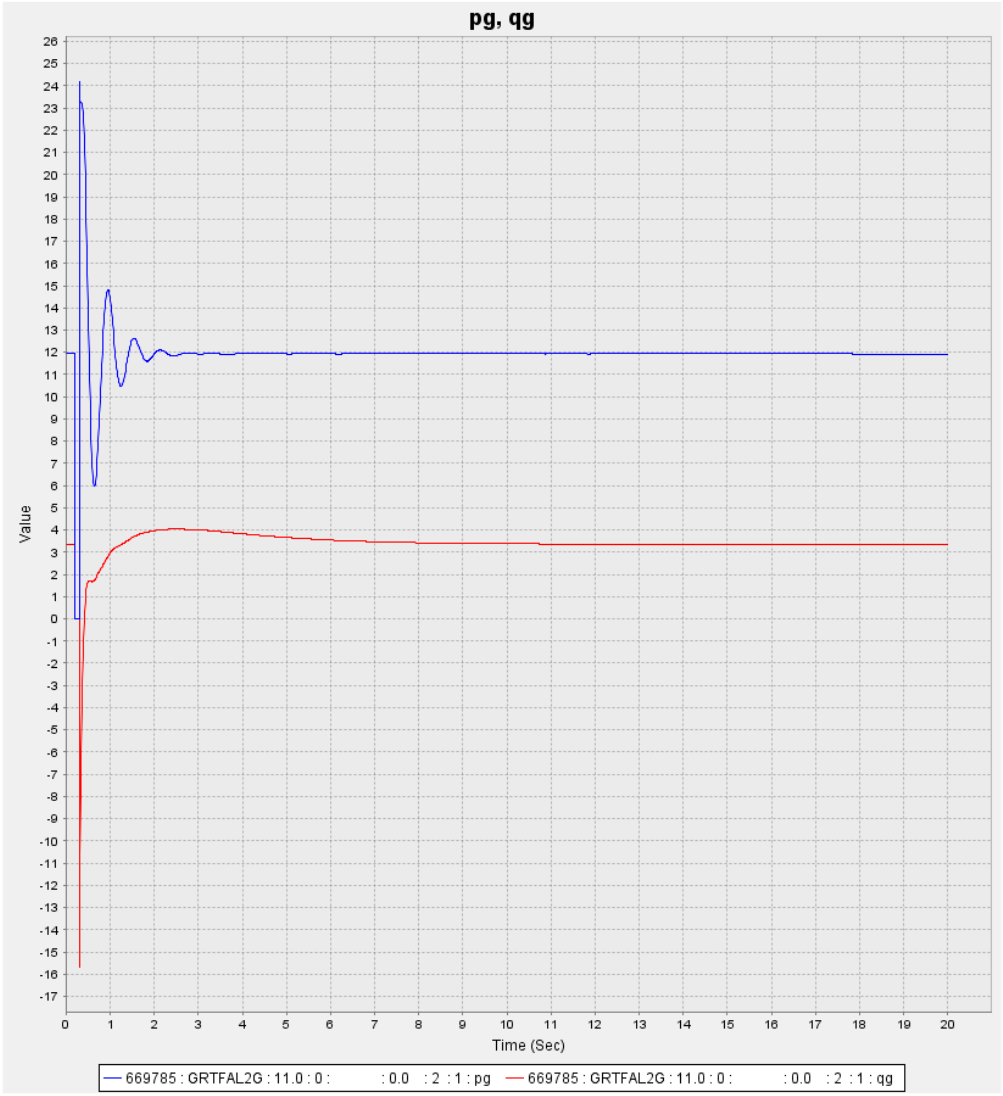



```
0,   100.00, 32, 0, 1, 60.00     / PSS®E-32.1    WED, JUL 03 2013   8:3
0,   100.00, 32, 0, 1, 60.00     / PSS®E-32.1    WED, JUL 03 2013   8:3
```

pslf.chf

Sat Dec 06 16:58:40 2014

Figure 54: 4 threads + new linear solver version of PSLF showing that the simulation accuracy remains unchanged

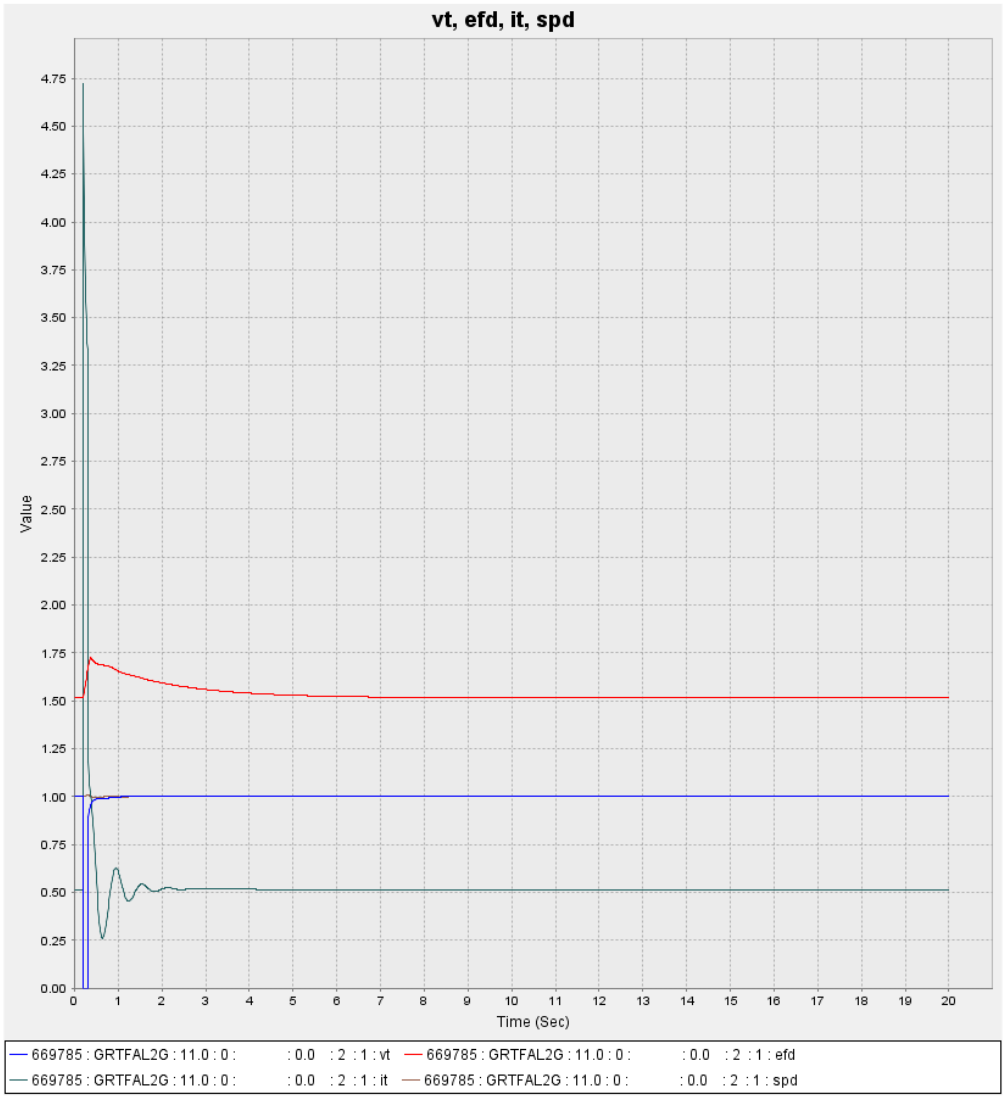



```
0,   100.00, 32, 0, 1, 60.00     / PSS®E-32.1     WED, JUL 03 2013   8:3
0,   100.00, 32, 0, 1, 60.00     / PSS®E-32.1     WED, JUL 03 2013   8:3
```

pslf.chf

Sat Dec 06 16:59:09 2014

Figure 55: 4 threads + new linear solver version of PSLF showing that the simulation accuracy remains unchanged

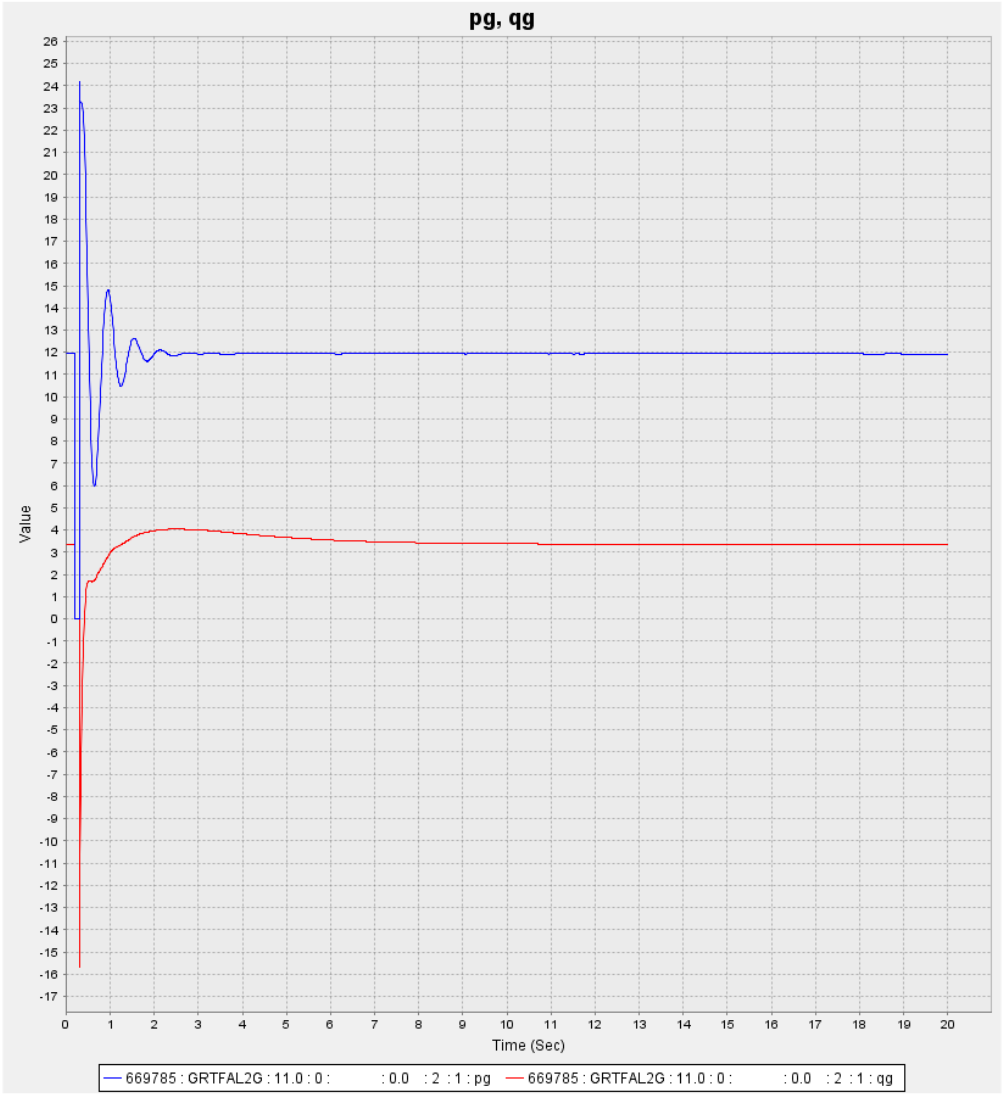



```
0,   100.00, 32, 0, 1, 60.00     / PSS®E-32.1    WED, JUL 03 2013   8:3
0,   100.00, 32, 0, 1, 60.00     / PSS®E-32.1    WED, JUL 03 2013   8:3
```

pslf.chf

Sat Dec 06 16:25:09 2014

Figure 56: 6 thread + new linear solver version of PSLF showing that the simulation accuracy remains unchanged



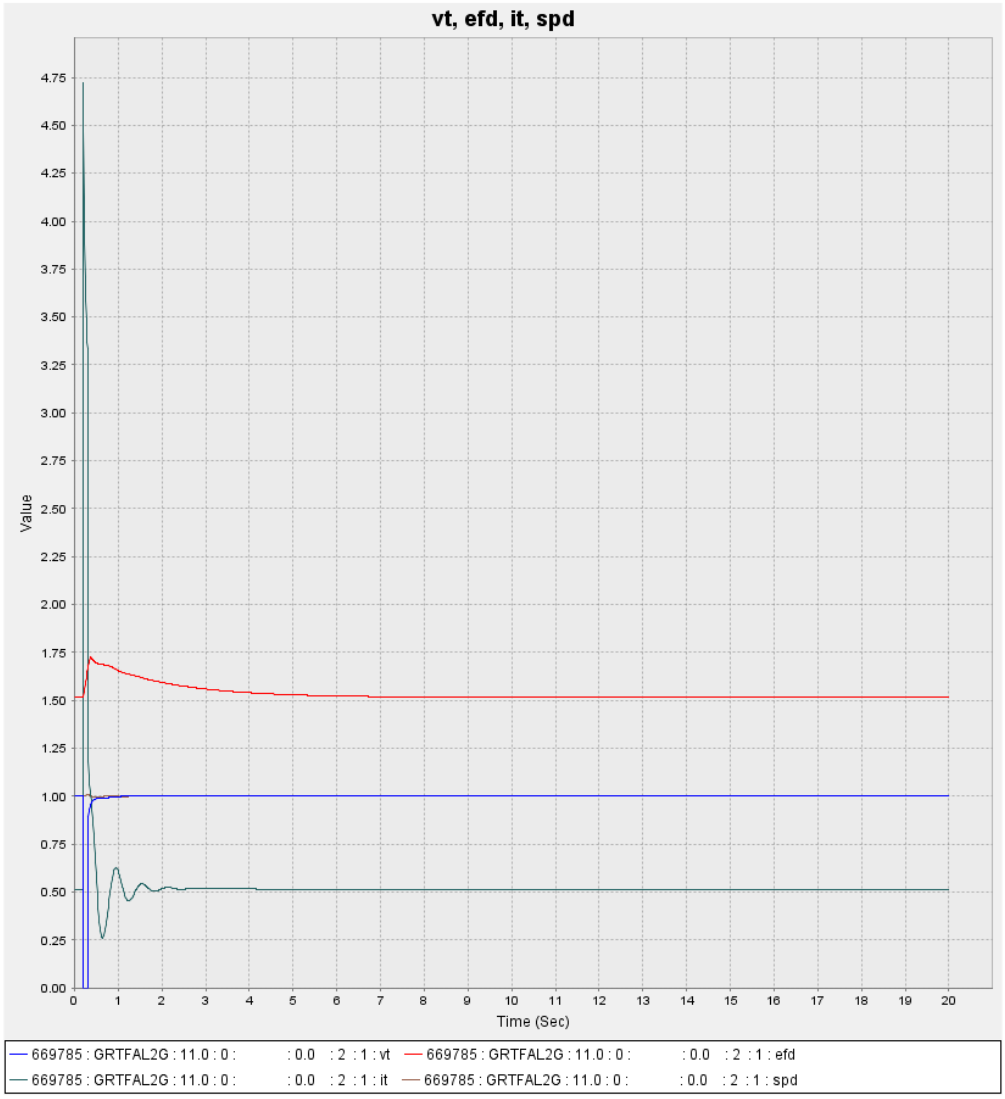

```
0,   100.00, 32, 0, 1, 60.00     / PSS®E-32.1     WED, JUL 03 2013   8:3
0,   100.00, 32, 0, 1, 60.00     / PSS®E-32.1     WED, JUL 03 2013   8:3
```

pslf.chf

Sat Dec 06 16:25:38 2014

Figure 57: 6 thread + new linear solver version of PSLF showing that the simulation accuracy remains unchanged

Appendix D – PSLF simulation results on PNNL's PIC machine



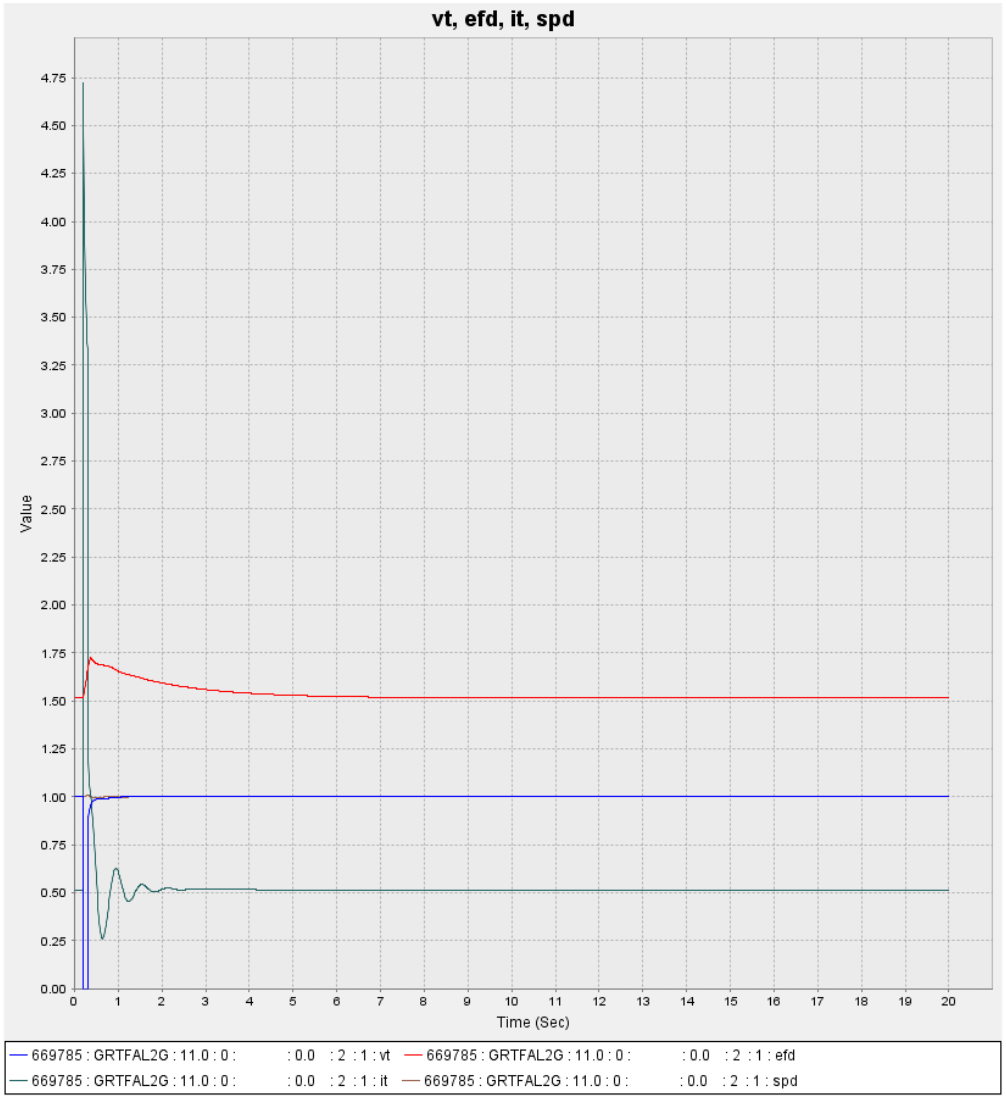

```
0,   100.00, 32, 0, 1, 60.00     / PSS®E-32.1     WED, JUL 03 2013   8:3
0,   100.00, 32, 0, 1, 60.00     / PSS®E-32.1     WED, JUL 03 2013   8:3
```

pslf20s_1c.chf

.

Mon Dec 08 18:35:08 2014

Figure 58: Single thread and native PSLF solver simulation showing that variable response remains unchanged



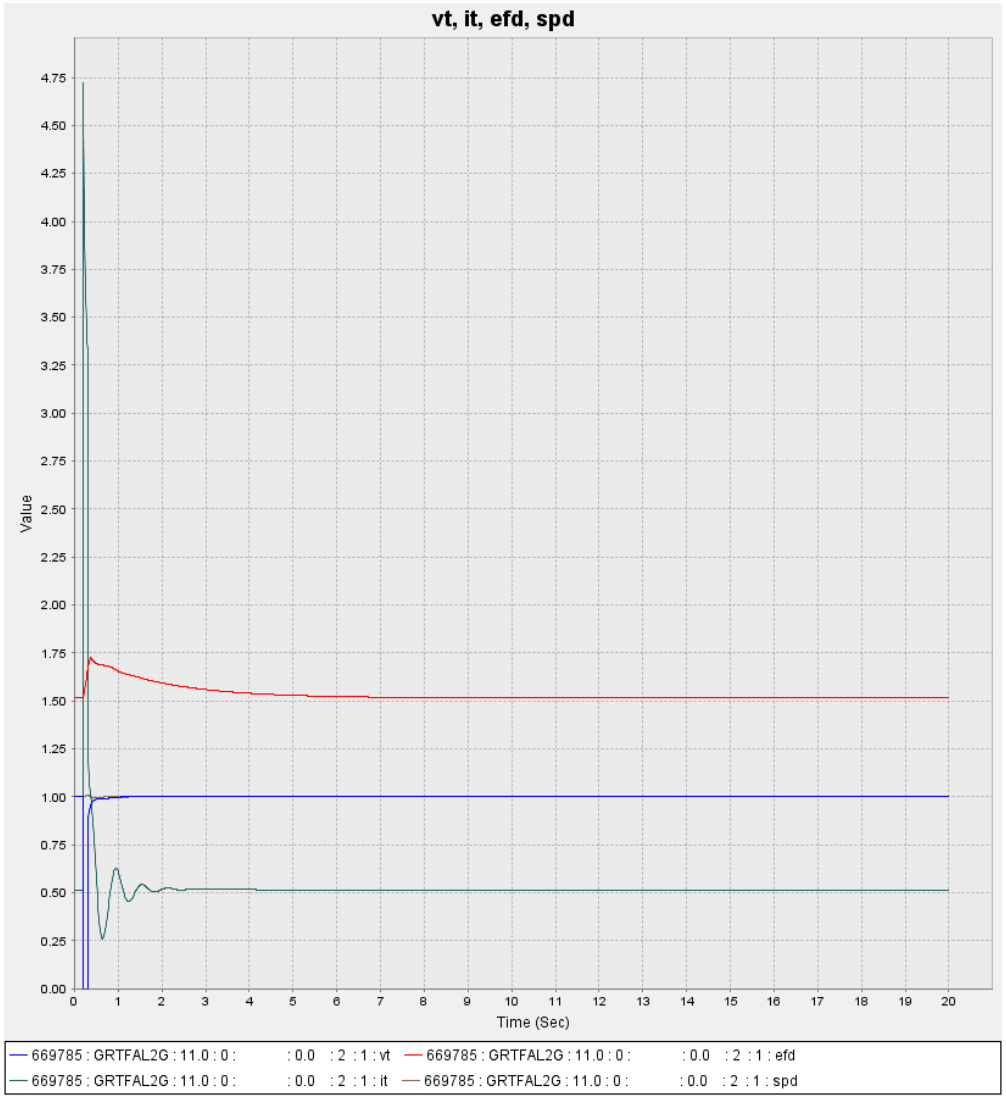

```
0,   100.00, 32, 0, 1, 60.00     / PSS®E-32.1     WED, JUL 03 2013   8:3
0,   100.00, 32, 0, 1, 60.00     / PSS®E-32.1     WED, JUL 03 2013   8:3
```

pslf20s_2c.chf

.

Mon Dec 08 18:51:39 2014

Figure 59: 2 threads + new linear system solver version of PSLF showing that simulation accuracy remains unchanged



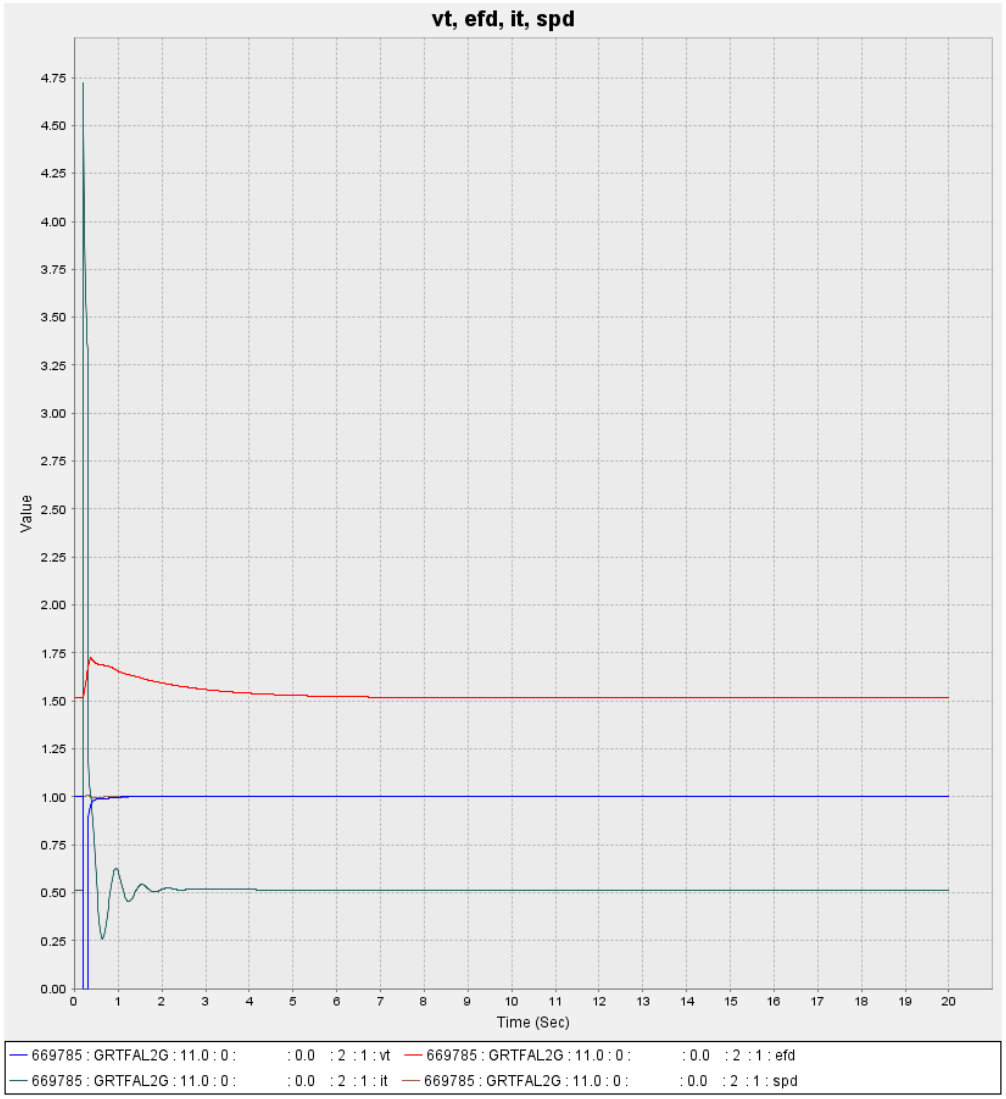

```
0,   100.00, 32, 0, 1, 60.00     / PSS®E-32.1     WED, JUL 03 2013   8:3
0,   100.00, 32, 0, 1, 60.00     / PSS®E-32.1     WED, JUL 03 2013   8:3
```

pslf20s_4c.chf

.

Mon Dec 08 19:08:10 2014

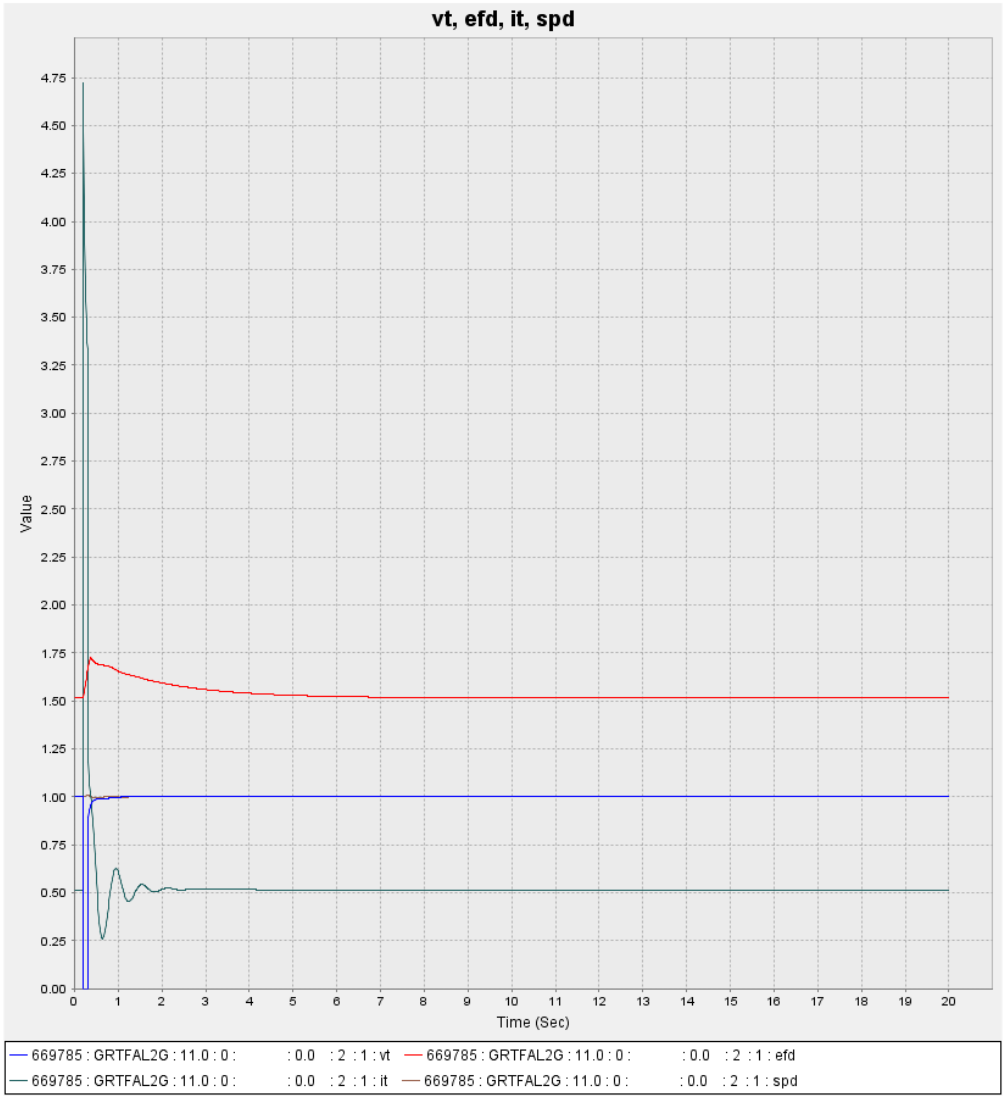
Figure 60: 4 threads + new linear system solver version of PSLF showing that simulation accuracy remains unchanged




```
0,   100.00, 32, 0, 1, 60.00     / PSS®E-32.1     WED, JUL 03 2013   8:3
0,   100.00, 32, 0, 1, 60.00     / PSS®E-32.1     WED, JUL 03 2013   8:3
```

pslf20s_6c.chf .
Mon Dec 08 19:21:29 2014

Figure 61: 6 threads + new linear system solver version of PSLF showing that simulation accuracy remains unchanged

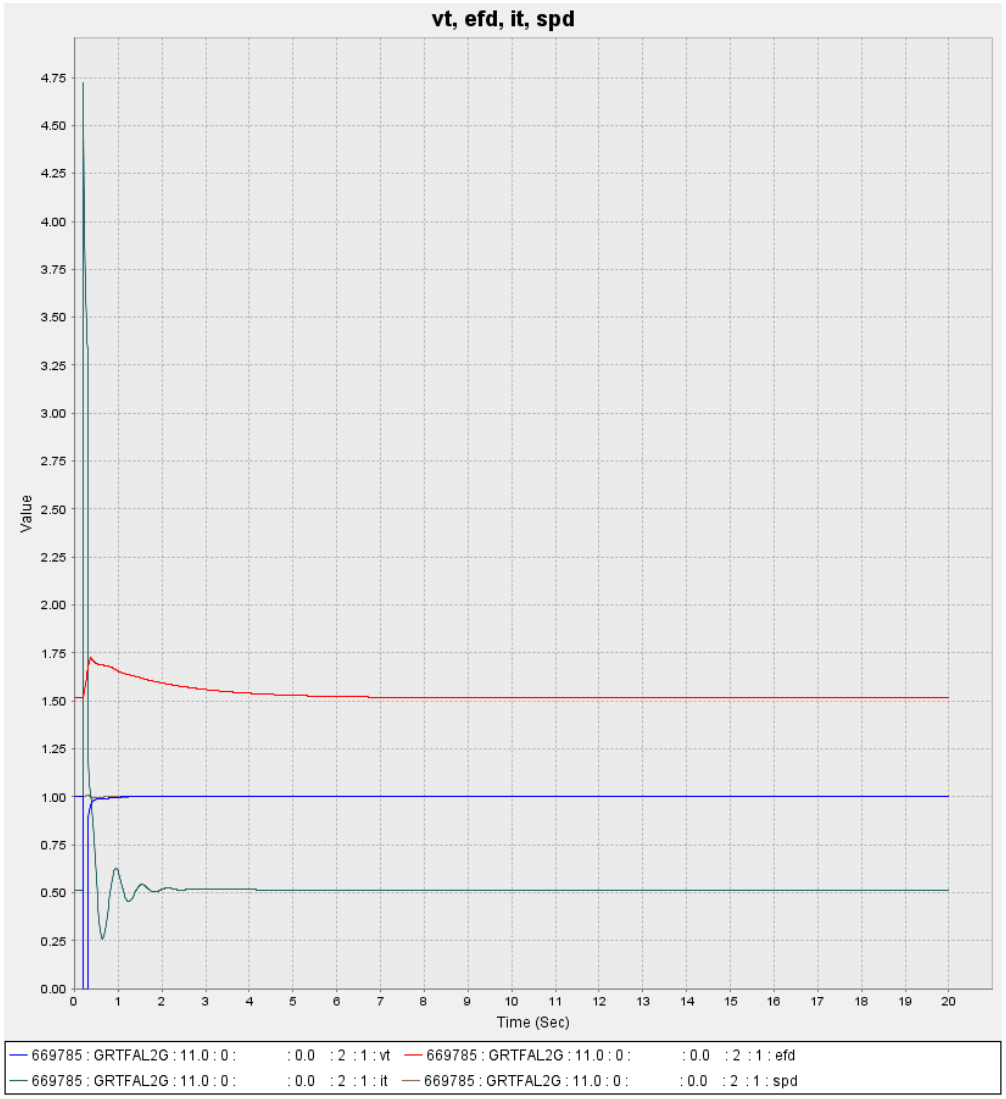



```
0,   100.00, 32, 0, 1, 60.00     / PSS®E-32.1     WED, JUL 03 2013   8:3
0,   100.00, 32, 0, 1, 60.00     / PSS®E-32.1     WED, JUL 03 2013   8:3
```

pslf20s_12c.chf .

Mon Dec 08 19:47:38 2014

Figure 62: 12 threads + new linear system solver version of PSLF showing that simulation accuracy remains unchanged

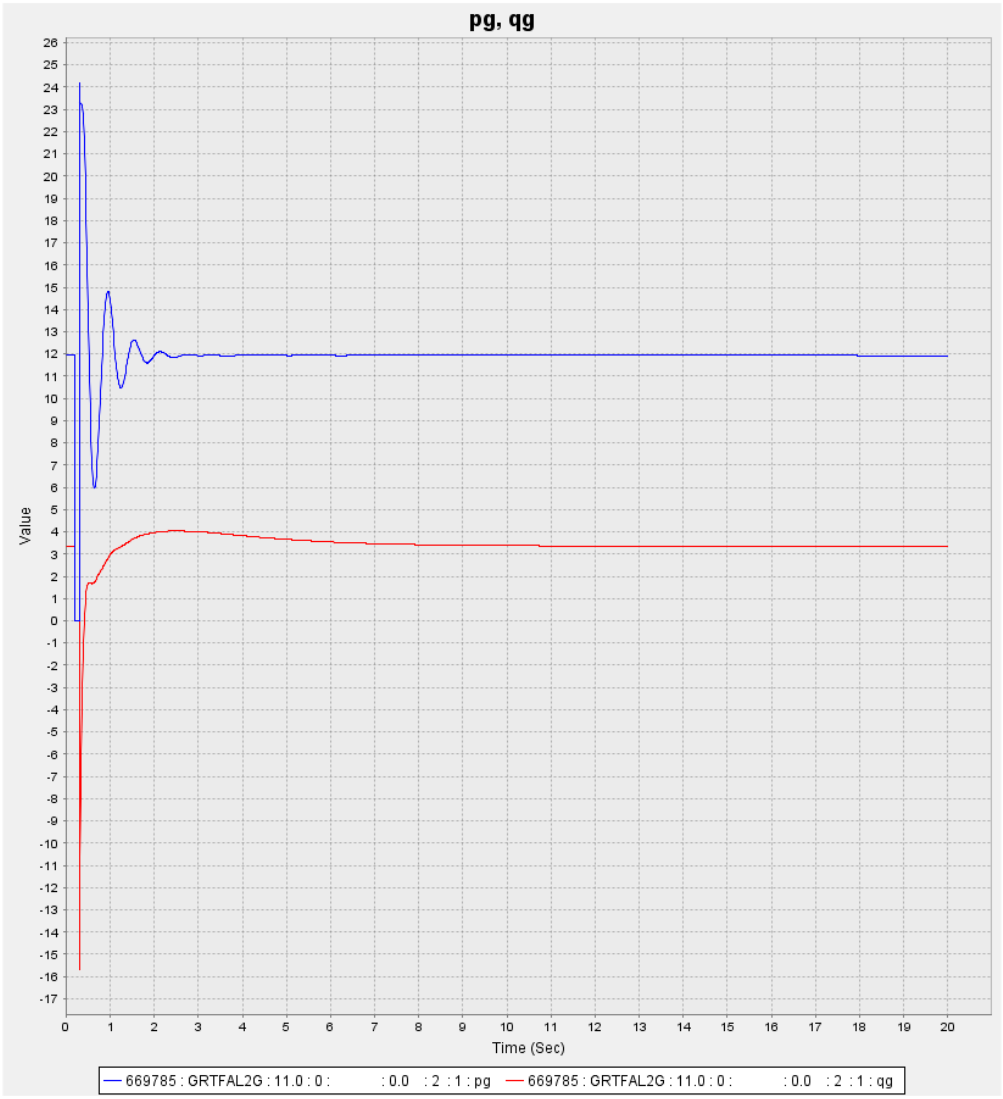



```
0,   100.00, 32, 0, 1, 60.00     / PSS®E-32.1     WED, JUL 03 2013   8:3
0,   100.00, 32, 0, 1, 60.00     / PSS®E-32.1     WED, JUL 03 2013   8:3
```

pslf20s_18c.chf .

Mon Dec 08 19:59:50 2014

Figure 63: 18 threads + new linear system solver version of PSLF showing that simulation accuracy remains unchanged



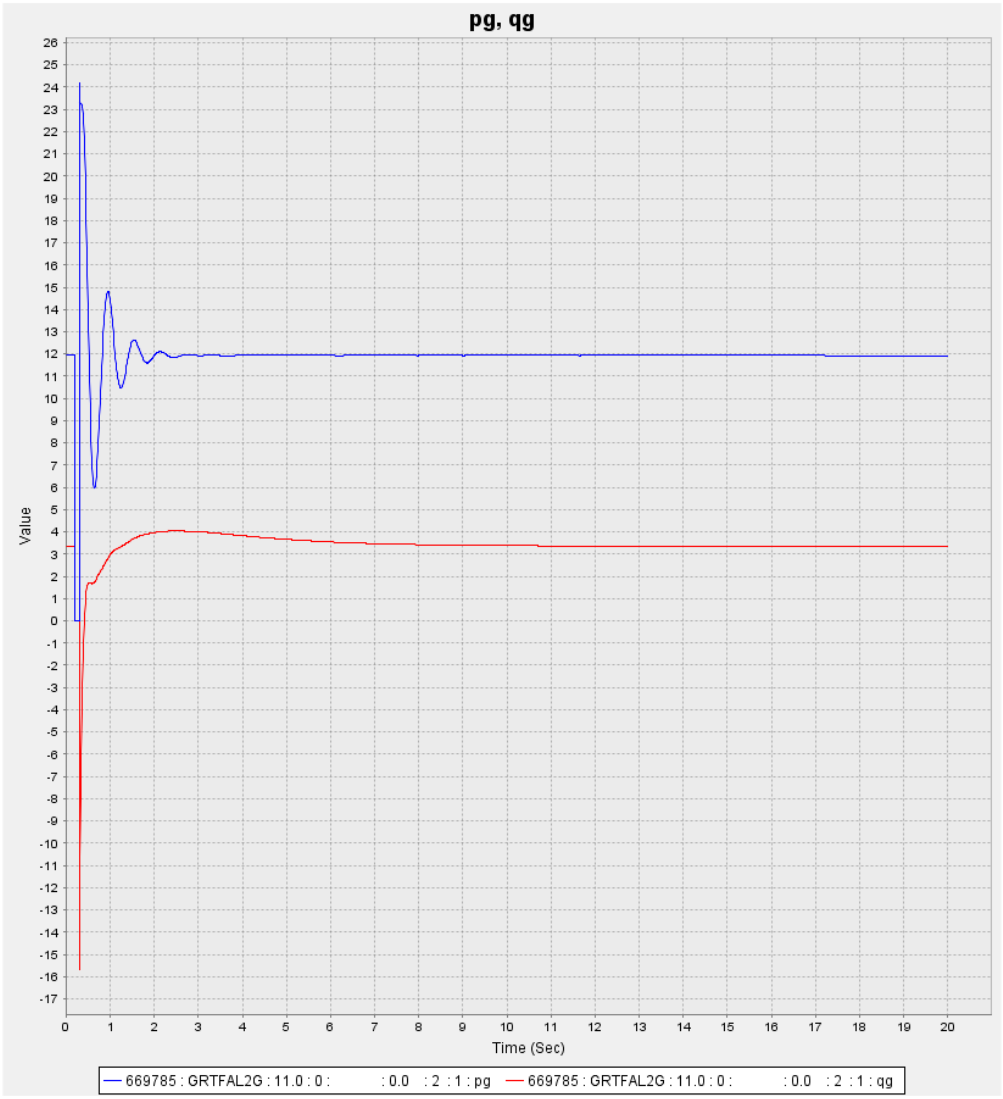

```
0,   100.00, 32, 0, 1, 60.00     / PSS®E-32.1     WED, JUL 03 2013   8:3
0,   100.00, 32, 0, 1, 60.00     / PSS®E-32.1     WED, JUL 03 2013   8:3
```

pslf20s_1c.chf

.

Mon Dec 08 18:35:54 2014

Figure 64: Single thread and native PSLF solver simulation showing that variable response remains unchanged



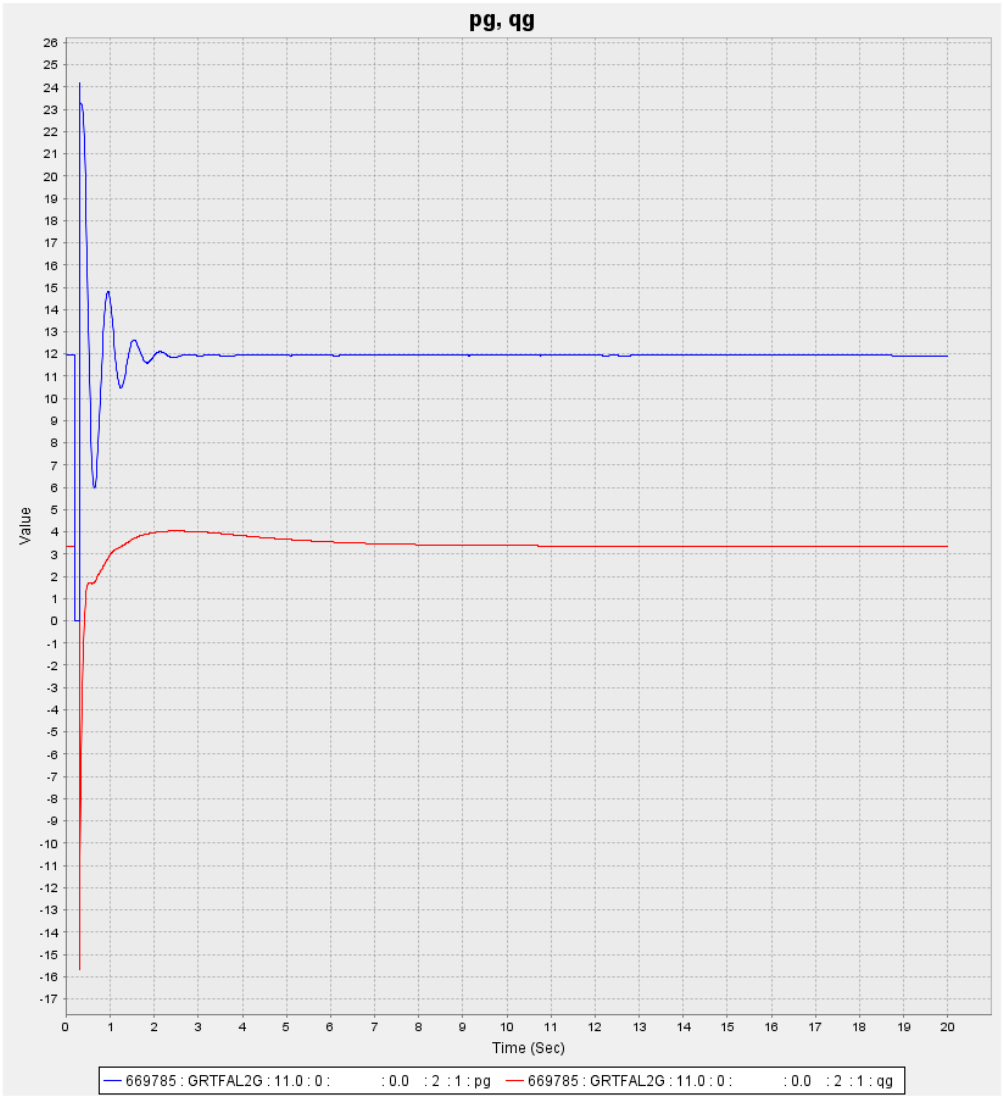

```
0,   100.00, 32, 0, 1, 60.00     / PSS®E-32.1     WED, JUL 03 2013   8:3
0,   100.00, 32, 0, 1, 60.00     / PSS®E-32.1     WED, JUL 03 2013   8:3
```

pslf20s_2c.chf

.

Mon Dec 08 18:52:23 2014

Figure 65: 2 threads + new linear system solver version of PSLF showing that simulation accuracy remains unchanged

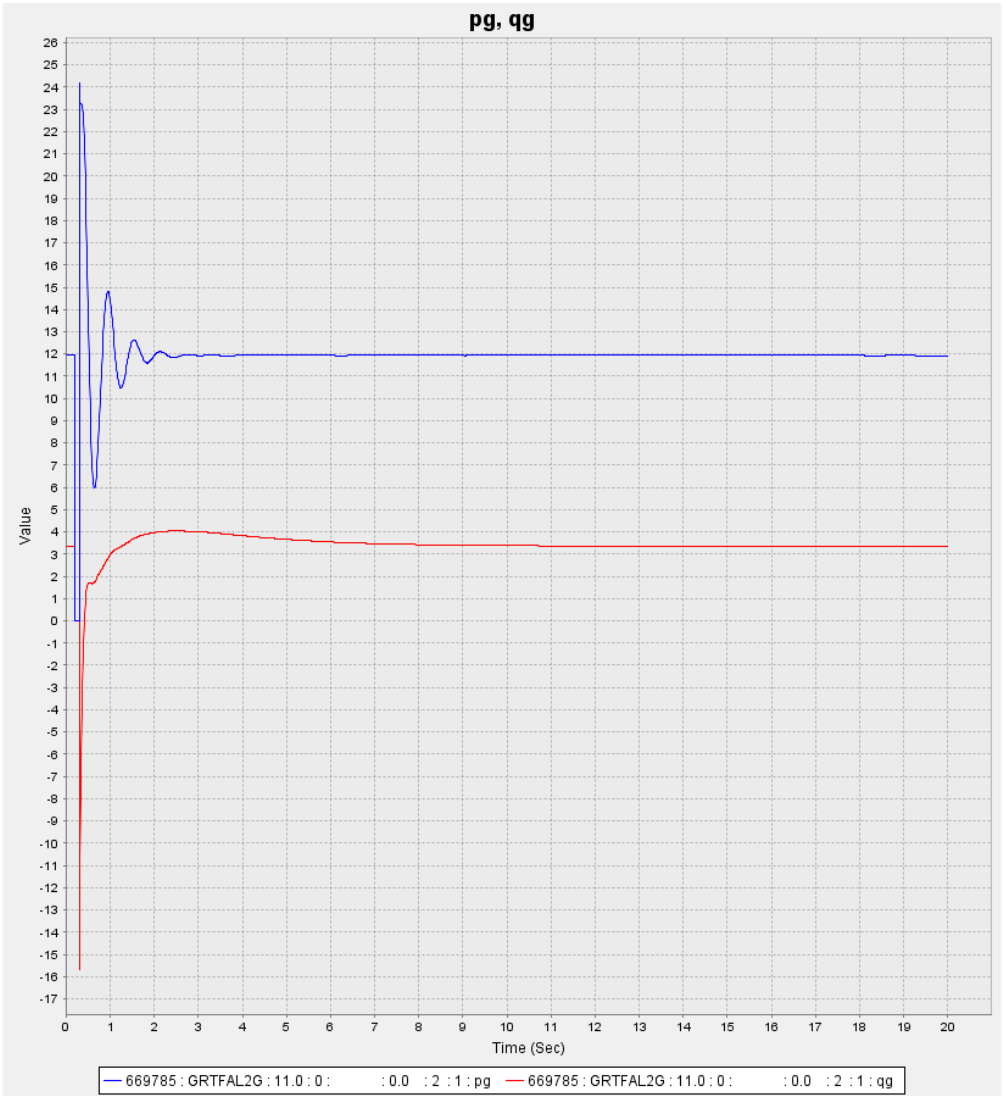



```
0,   100.00, 32, 0, 1, 60.00     / PSS®E-32.1     WED, JUL 03 2013   8:3
0,   100.00, 32, 0, 1, 60.00     / PSS®E-32.1     WED, JUL 03 2013   8:3
```

pslf20s_4c.chf .

Mon Dec 08 19:08:46 2014

Figure 66: 4 threads + new linear system solver version of PSLF showing that simulation accuracy remains unchanged



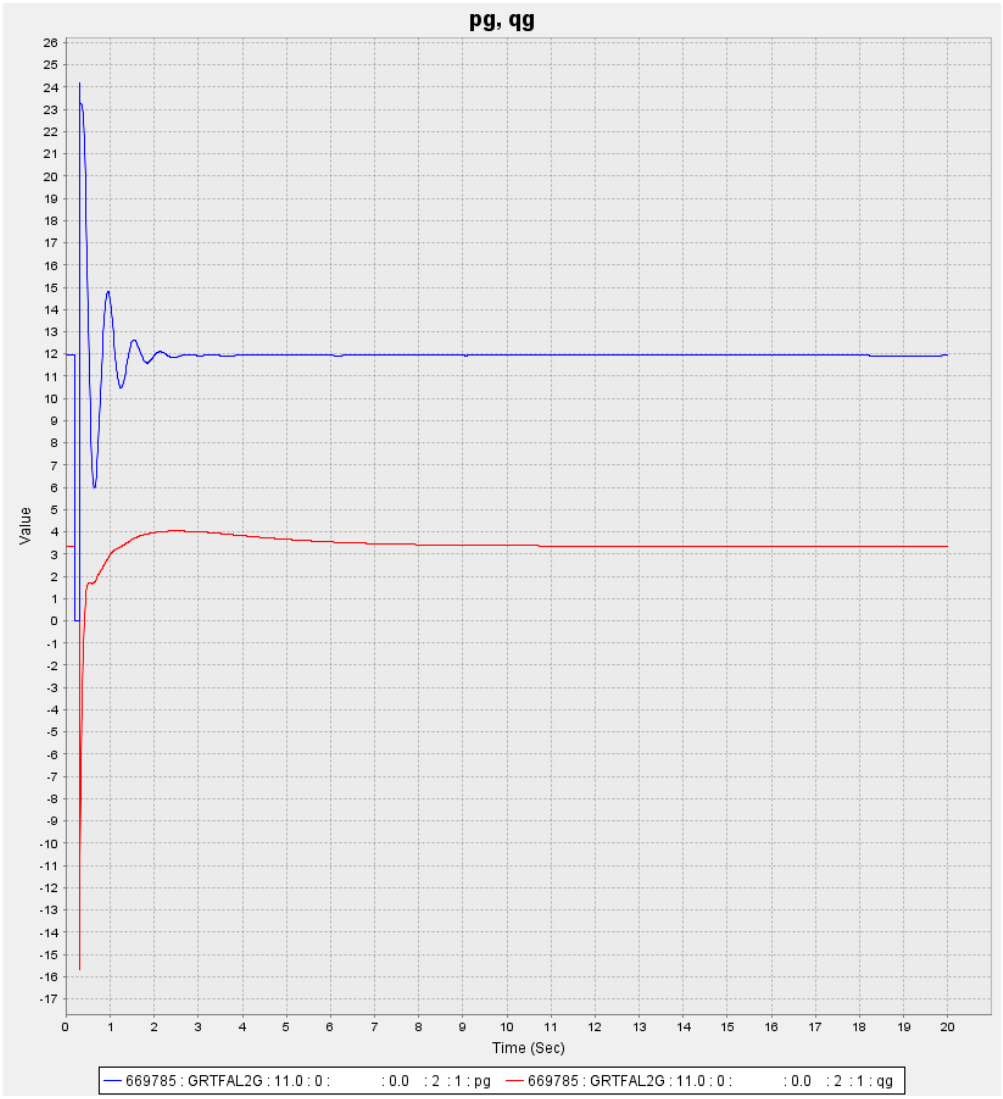

```
0,   100.00, 32, 0, 1, 60.00     / PSS®E-32.1     WED, JUL 03 2013   8:3
0,   100.00, 32, 0, 1, 60.00     / PSS®E-32.1     WED, JUL 03 2013   8:3
```

pslf20s_6c.chf

.

Mon Dec 08 19:22:36 2014

Figure 67: 6 threads + new linear system solver version of PSLF showing that simulation accuracy remains unchanged



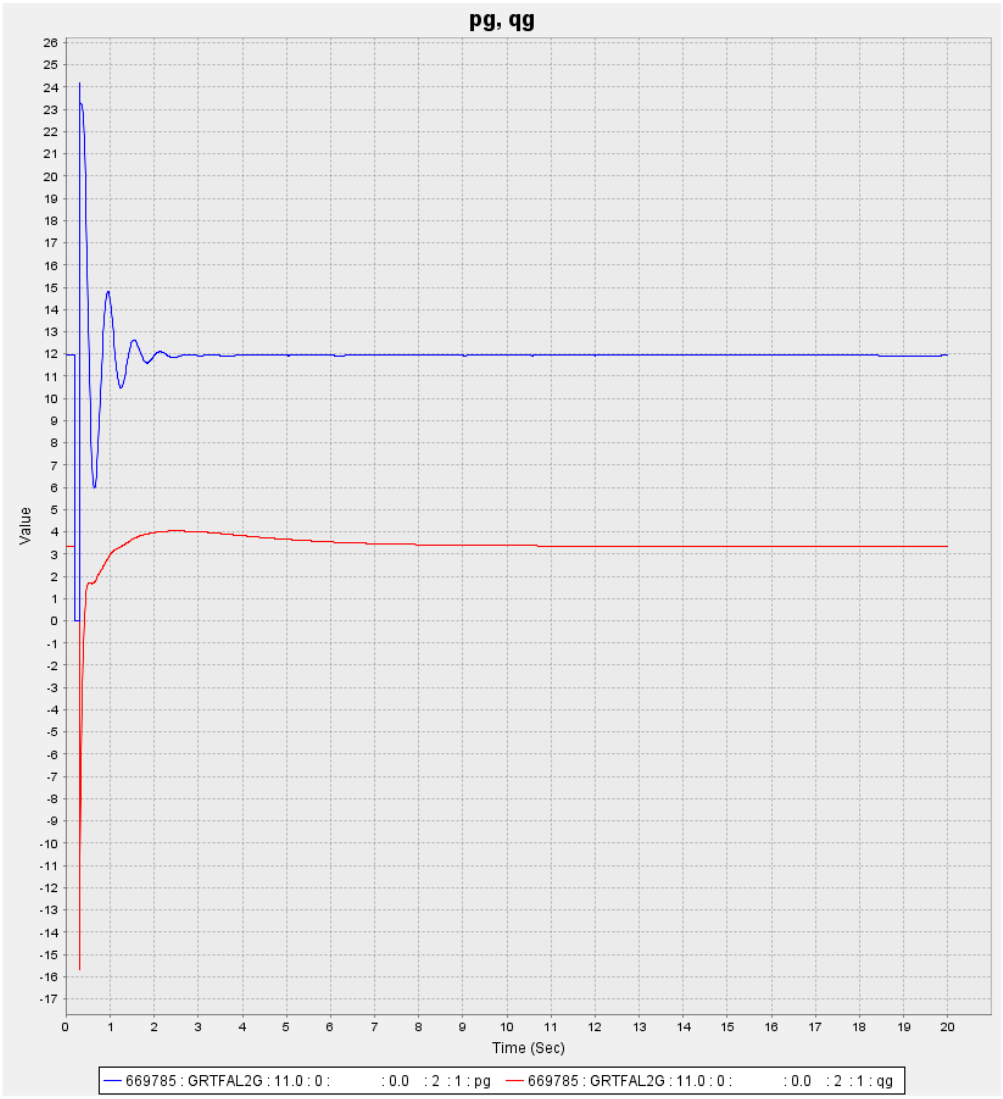

```
0,   100.00, 32, 0, 1, 60.00     / PSS®E-32.1    WED, JUL 03 2013   8:3
0,   100.00, 32, 0, 1, 60.00     / PSS®E-32.1    WED, JUL 03 2013   8:3
```

C:\Data\Projects\DOE FOA 729\PNNL_PIC_RUNS\figs\PIC runs\PIC\chfs\pslf20s_12c.chf

s\PIC\chfs

Wed Dec 10 15:57:28 2014

Figure 68: 12 threads + new linear system solver version of PSLF showing that simulation accuracy remains unchanged




```
0,   100.00, 32, 0, 1, 60.00     / PSS®E-32.1     WED, JUL 03 2013   8:3
0,   100.00, 32, 0, 1, 60.00     / PSS®E-32.1     WED, JUL 03 2013   8:3
```

pslf20s_18c.chf .
Mon Dec 08 20:00:47 2014

Figure 69: 18 threads + new linear system solver version of PSLF showing that simulation accuracy remains unchanged