

Real-Time Power System Dynamics Simulation Using a Parallel Block-Jacobi Preconditioned Newton-GMRES Scheme

Shrirang Abhyankar *Member, IEEE*, Alexander J. Flueck *Senior Member, IEEE*

Abstract—Real-time dynamics simulation of large-scale power systems is a computational challenge because of the need to solve a large set of stiff, nonlinear differential-algebraic equations. The main bottleneck in these simulations is the solution of the linear system during each nonlinear iteration of Newton’s method. We present a parallel linear solution scheme using the Krylov subspace-based iterative solver GMRES with a Block-Jacobi preconditioner. The scheme shows promise for real-time dynamics simulation, with a good speed up for a 2383-bus, 327-generator test case. Results obtained for both stable and unstable operating conditions show that real-time simulation speed can be realized by using the proposed parallel linear solution scheme.

Index Terms—Transient Stability, Parallel Computing, Block-Jacobi Preconditioner, Newton-GMRES, PETSc.

I. INTRODUCTION

THE need for faster, and accurate, power grid dynamics simulation (or transient stability analysis) has been a primary focus of the power system community in recent years. This view was reiterated in the recent DOE and EPRI workshops [14], [20]. Indeed, more than two decades ago, real-time dynamics simulation was identified as a “grand computing challenge” [23]. As processor speeds were increasing, real-time dynamics simulation appeared possible in the not-too-distant future. Unfortunately, processor clock speeds saturated about a decade ago, and real-time dynamics simulation remains a grand computing challenge.

Dynamics simulation of a large-scale power system is computationally challenging because of the presence of a large set of stiff, nonlinear differential-algebraic equations (DAEs), where the differential equations model dynamics of the rotating machines (e.g., generators and motors) and the algebraic equations represent the transmission system and quasi-static loads. The electrical power system is expressed as a set of nonlinear DAEs, where f and g are vector-valued nonlinear functions.

$$\begin{aligned} \frac{dx}{dt} &= f(x, y) \\ 0 &= g(x, y) \end{aligned} \quad (1)$$

The solution of the dynamic model given in (1) needs the following:

Shrirang Abhyankar (e-mail: abhyshr@mcs.anl.gov) is with the Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439.

A. J. Flueck (e-mail: flueck@iit.edu) is with the Department of Electrical and Computer Engineering, Illinois Institute of Technology, Chicago, IL, 60616.

- A numerical integration scheme to convert the differential equations in algebraic form
- A nonlinear solution scheme to solve the resultant nonlinear algebraic equations
- A linear solver to solve the update step at each iteration of the nonlinear solution

Figure 1 shows the wall-clock execution time of a series of dynamics simulations on a single processor for a temporary three-phase fault applied for 0.1 seconds.

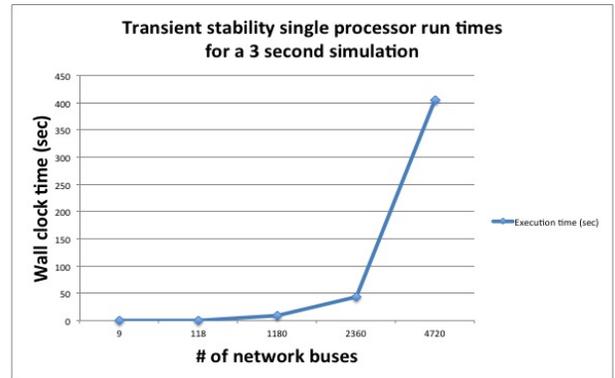


Fig. 1. Single processor dynamic simulation execution times for a 3 second simulation period on different systems for 0.1 second three-phase balanced temporary fault.

The test cases with bus sizes greater than 1000 were obtained by duplicating the 118 bus test system 10, 20, and 40 times respectively, and connecting each 118 bus area by five randomly chosen tie lines. As system size increases, execution time grows dramatically. Thus “real-time” dynamics analysis of a utility or a regional operator network is an enormous computing challenge.

For example, PJM, a regional transmission organization (RTO) covering 168,500 square miles of 12 different states, monitors approximately 13,500 buses [21]. Similarly, the Electric Reliability Council of Texas (ERCOT) monitors approximately 18,000 buses [13]. High-level Eastern Interconnection models contain more than 50,000 buses. To perform dynamics simulation in real time, the simulator must compute the solution to a set of equations containing more than 150,000 variables in a few milliseconds. Because of this high computational cost, dynamics analysis is usually performed on relatively small interconnected power system models, and computation is mainly performed offline. Researchers at Pacific Northwest National Laboratory have reported that a

simulation of 30 seconds of dynamic behavior of the Western Interconnection requires about 10 minutes of computation time today on an optimized single processor [17].

II. SPEEDING POWER GRID DYNAMICS SIMULATION BY PARALLEL COMPUTING

A natural way to speed this computation is to use parallel computing techniques, namely, share the computational load amongst multiple processors. The need for parallelizing existing power system applications is even greater as the computer hardware industry moves toward multicore and many core architectures. All major computer vendors are aggressively introducing a new generation of hardware that incorporates multiple cores on a chip, sometimes with additional simultaneous multithreading capabilities. Products incorporating 6 and 8 cores are already on the market. The number of cores per chip is expected to grow rapidly, so that even in the relatively short term, a single chip is expected to support the execution of a few hundred threads. These multicore architectures can be utilized efficiently only with parallel algorithms that distribute the computational load over multiple cores. Several workshops [14], [20] have highlighted the need for investigating these multicore/many-core architectures for accelerating performance of power system applications.

In the context of parallel algorithms for dynamics simulations, most of the research effort was done over the past decade. The parallel-in-space algorithms partition the given network into loosely coupled or independent subnetworks. Each processor is then assigned equations for a subnetwork. The partitioning strategy for the network division is critical for parallel-in-space algorithms to minimize the coupling between the subnetworks, that is, to reduce the inter processor communication, and balance the work load. Once a suitable partitioning strategy is selected the next key step is the solution of the linear system in each Newton iteration. Several linear solution schemes have been proposed in the literature, of which the most prominent are the *very dishonest newton method* and *conjugate gradient* method. Reference [9] uses the very dishonest Newton method in which the factorization of the Jacobian matrix is done only when a certain fixed number of iterations are exceeded. Reference [11] decomposes the network equations in a block bordered diagonal form (BBDF) and then uses a hybrid solution scheme using LU and conjugate gradient. Reference [12] solves the network by a block-parallel version of the preconditioned conjugate gradient method. The network matrix in [12] is in the near block diagonal form (NBDF).

The parallel-in-time approach was introduced in [5]. The idea of this approach seeks to combine the differential and algebraic equations over several time steps, create a bigger system, and solve them simultaneously using the Newton method. All the equations for several integration steps are assigned to each processor.

Waveform relaxation methods [18], [10], [16] involve a hybrid scheme of space and time parallelization in which the network is partitioned in space into subsystems and then distributed to the processors. Several integration steps for each

subsystem are solved independently to get a first approximation [15]. The results are then exchanged, and the process is repeated. The advantage of this scheme is that each subsystem can use a different integration step and/or different integration algorithm (multirate integration).

III. PARALLEL-IN-SPACE DYNAMICS SIMULATION APPROACH

This section describes the parallel implementation of our dynamics simulator. Unlike other dynamics simulators that use a per phase balanced network model, our simulator is a three-phase dynamics simulator. The details of the three-phase dynamics simulator can be found in [1], [2]. The simulator is developed using the mathematical and computing platform of the high performance library PETSc [8].

A. Domain Decomposition

We adopt a domain decomposition approach that partitions the power system network into several subnetworks. Figure 2 shows an illustrative example of the division of the IEEE 118-bus system into two subnetworks. Each subnetwork is then

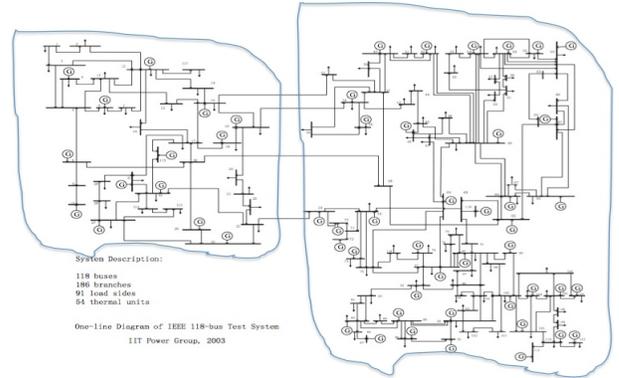


Fig. 2. Partitioning of the IEEE 118 bus system for 2 processors

the domain of operation of a processor, in other words, each processor is assigned the DAE equations for the subnetwork. Equation (2) represents the equations for each processor, where the subscript p represents the variables for the current processor, and the subscript c represents the variables needed from other processors to compute the function on the current processor.

$$\begin{aligned} \frac{dx_p}{dt} &= f(x_p, y_p, y_c) \\ 0 &= g(x_p, y_p, y_c) \end{aligned} \quad (2)$$

Note that the differential equations i.e. the electromechanical machine equations are naturally decoupled because they are incident at a bus only, whereas the algebraic network equations require communication with other processors to compute their local function. Hence the partitioning of the network is done by using only the topology of the network. The partitioning of the network can be done by hand via judicious topology scanning or by graph partitioning techniques.

For our simulator, we use the ParMetis package [22] for doing the network partitioning. ParMetis is a parallel graph partitioning package that is used for partitioning of unstructured

graphs. It tries to partition a given graph with the objective of minimizing the edge cuts while having balanced partitions, that is, balancing computational load for each processor.

The ParMetis package requires an *adjacency matrix* whose elements are 1s and 0s, where an element $A_{i,j}$ is 1 if vertex i is connected to vertex j . Along with the *adjacency matrix*, a weight can be assigned to each vertex to account for different computational requirements. With vertex weights, ParMetis tries to minimize the edge cuts and also have the same number of vertex weights in each subdomain.

For our simulator, the connection information from the single-phase Y_{bus} matrix of the network was used as the adjacency graph for ParMetis. Larger weights were assigned to the vertices having generators to account for the extra computation involved for the generator differential and algebraic equations.

B. Generalized Minimum Residual Method (GMRES)

Newton's method requires the solution of the linear system

$$J(x^i)\Delta x^{i+1} = -F(x^i), \quad (3)$$

where i is the iteration count. Solution of (3) can be done by either direct or iterative methods. Krylov subspace iterative methods are the most popular among the iterative methods for solving large linear systems. These methods are based on projection onto subspaces called Krylov subspaces of the form b, Ab, A^2b, A^3b, \dots . A general projection method for solving the linear system

$$Ax = b \quad (4)$$

is a method that seeks an approximate solution x_m from an affine subspace $x_0 + K_m$ of dimension m by imposing

$$b - Ax_m \perp L_m$$

where L_m is another subspace of dimension m and x_0 is an arbitrary initial guess to the solution. A Krylov subspace method is a method for which the subspace K_m is the Krylov subspace

$$K_m(A, r_0) = \text{span}\{r_0, Ar_0, A^2r_0, A^3r_0, \dots, A^{m-1}r_0\},$$

where $r_0 = b - Ax_0$. The different versions of Krylov subspace methods arise from different choices of the subspace L_m and from the ways in which the system is preconditioned.

The generalized minimum residual method (GMRES)[25] is a projection method based on taking $L_m = AK_m(A, r_0)$ in which K_m is the m th Krylov subspace. This technique minimizes the residual norm over all vectors $x \in x_0 + K_m$. In particular, GMRES creates a sequence x_m that minimizes the norm of the residual at step m over the m th Krylov subspace

$$\|b - Ax_m\|_2 = \min\|b - Ax\|_2 \quad (5)$$

At step m , an Arnoldi process is applied for the m th Krylov subspace in order to generate the next basis vector. When the norm of the new basis vector is sufficiently small, GMRES solves the minimization problem

$$y_m = \text{argmin}\|\beta e_1 - \bar{H}_m y\|_2,$$

where \bar{H}_m is the $(m+1) \times m$ upper Hessenberg matrix.

C. Block-Jacobi Preconditioner

The convergence of the Krylov subspace linear solvers depends on the eigenvalues of the operating matrix A and can be slow if the matrix has widely dispersed eigenvalues, such as ill-conditioned power system matrices. Hence, to speed up the convergence, a preconditioner matrix M^{-1} , where M^{-1} approximates A^{-1} , is generally used. A *preconditioner* is a matrix that transforms the linear system

$$Ax = b$$

into another system with a better spectral properties for the iterative solver. If M is the preconditioner matrix, then the transformed linear system is

$$M^{-1}Ax = M^{-1}b. \quad (6)$$

Equation (6) is referred to as being preconditioned from the left, but one can also precondition from the right

$$AM^{-1}y = b, \quad x = M^{-1}y, \quad (7)$$

or split preconditioning

$$M_1^{-1}AM_2^{-1}y = M_1^{-1}b, \quad x = M^{-1}y, \quad (8)$$

where the preconditioner is $M = M_1M_2$.

When Krylov subspace methods are used, it is not necessary to form the preconditioned matrices $M^{-1}A$ or AM^{-1} explicitly since this is an expensive process. Instead, matrix-vector products with A and solutions of linear systems of the form $Mz = r$ are performed (or matrix-vector products with M^{-1} if explicitly known).

Designing a good preconditioner depends on the choice of iterative method, problem characteristics, and so forth. In general a good preconditioner should be (a) cheap to construct and apply, and (b) the preconditioned system should be easy to solve.

With the Jacobian matrix in a nearly bordered block diagonal form, the diagonal block on each processor can be used as a preconditioner. As an example, if the Jacobian matrix is distributed to two processors (0 and 1) as follows

$$\begin{bmatrix} 0 \\ 1 \end{bmatrix} \begin{bmatrix} J_1 & J_2 \\ J_3 & J_4 \end{bmatrix},$$

then the parallel Block-Jacobi preconditioner is

$$\begin{bmatrix} 0 \\ 1 \end{bmatrix} \begin{bmatrix} J_1^{-1} & \\ & J_4^{-1} \end{bmatrix}.$$

The factorization of J_1 and J_4 can be done independently on each processor, and no communication is required for building the preconditioner.

D. Matrix Reordering

By reordering the rows and columns of a matrix, it may be possible to reduce the amount of fill-in created by LU factorization, thereby decreasing the number of floating point operations and storage. We experimented with various reordering strategies, available with the Portable Extensible Toolkit for Scientific Computation [7] on the test systems to determine the optimal reordering strategy, namely, the ordering scheme

resulting in the least number of nonzeros in the factored matrix. The *quotient minimum degree* ordering was found to be the best approach for the systems that we tested.

IV. SCALABILITY RESULTS

The 2383-bus system provided with the MatPower [27] package distribution was used to test the scalability of the simulator. This test case has 327 generators and 2896 branches supplying a total load of 25281 MW. This case represents the Polish 400, 220, and 110 kV networks during winter 1999-2000 peak conditions. For the dynamic simulations, all the generators were modeled by using the GENROU model [19] with an IEEE Type 1 exciter model [19], and the loads modeled as constant impedances. The numerical integration scheme used is an implicit trapezoidal scheme with a time step of 0.01667 seconds.

The parallel performance runs were done on a shared-memory machine with four 2.2 GHz AMD Opteron 6274 processors. Each processor has 16 cores, giving a total of 64 cores. The code for the developed simulator is written in C using the PETSc library framework and compiled with GNU's gcc compiler with -O3 optimization.

Since our goal is realizing a real-time dynamics simulation, we define the metric "real-time speedup factor" s_r given in (9) in order to assess the proximity of the simulation to real-time. A value of $s_r \geq 1$ indicates that the simulation is running in realtime or faster than realtime.

$$s_r = \frac{T_s}{T_e} \quad (9)$$

T_s is the simulation time length, and T_e is the simulation execution time.

In the following subsections we present the scalability of our dynamics simulator using a Block-Jacobi (1 block/core) preconditioned GMRES scheme and compare it with parallel sparse LU factorization using the MUMPS [6] package. PETSc provides parallel GMRES as the default linear solver and the choice of variety of preconditioners including Block-Jacobi with ILU(0) and nested dissection reordering which is the default parallel preconditioner. We use LU with a quotient minimum degree reordering for the Block-Jacobi preconditioner. MUMPS is a parallel sparse direct linear solver that uses a multifrontal approach[6] for the parallel solution of $Ax = b$.

Two cases are considered for assessing the scalability: (1) A three-phase fault on bus 185 for 0.1 seconds that results in stable dynamics and (2) a three-phase fault on bus 18 for 0.1 seconds that results in unstable dynamics. The dynamics of the 2383 bus system were simulated for 3 seconds.

A. Stable Case: Three-phase Fault on Bus 185

Figure 3 shows the dynamics of generator speeds for a three-phase fault on bus 185 for from $t=0.0$ sec to $t=0.1$ sec. Bus 185 has a large load of 362 MW. Following the fault, the generators depart from their synchronous mode of operation but quickly regain synchronism, as seen in Figure 3.

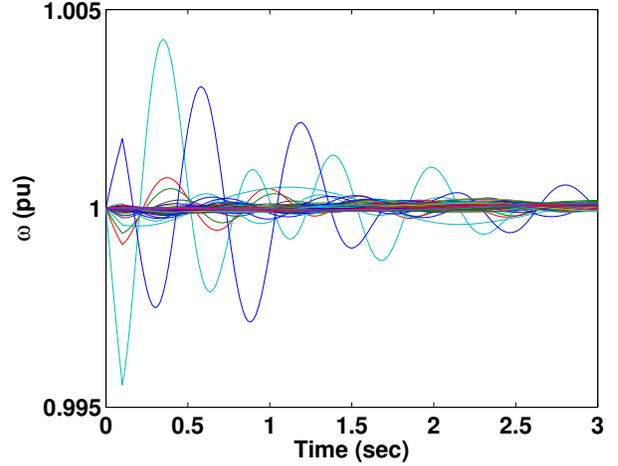


Fig. 3. Generator speeds for a three-phase fault applied for 0.1 seconds on bus 185

Figures 4 and 5 show the execution times and the real-time speedup factor s_r for the stable operating conditions. As seen, the Block-Jacobi Newton-GMRES scheme shows significant speedup as compared with a parallel LU factorization using MUMPS.

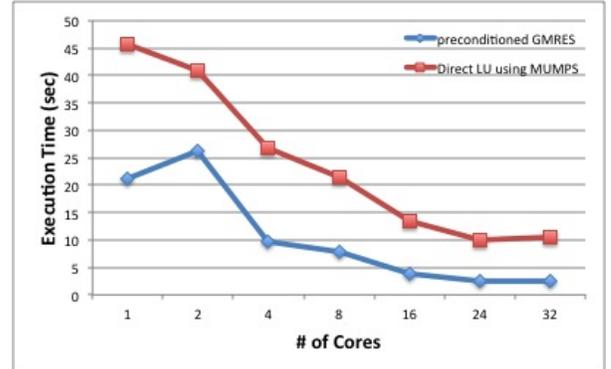


Fig. 4. Execution times for stable 2383 bus system dynamics

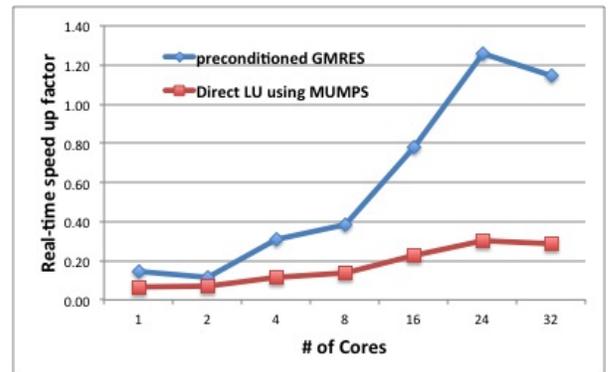


Fig. 5. Real-time speedup factor for stable 2383 bus system dynamics

B. Unstable Case: Three-phase Fault on Bus 18

For testing the unstable dynamics, a three-phase fault was placed on Bus 18 from $t = 0.0$ sec to $t = 0.1$ sec. The generator with the largest real power output is connected to Bus 18. As seen from Figure 6, following the fault the generator on bus 18 loses synchronism, and its speed drops. The unstable dynamics cause an increased number of nonlinear and linear iterations due to increased swinging, or separation, of the generator rotor angles. For the stable case, the maximum number of Newton iterations observed was 3, while for the unstable case it was 5. The execution time on 1 processor using the Block-Jacobi preconditioned GMRES scheme takes 21.21 seconds for the stable case while it takes 23.38 seconds for the unstable case. However, on 24 cores the simulation execution requires only slightly more than 3 seconds of simulated time. Thus, real-time simulation is nearly achieved on 24 cores with the proposed scheme. In comparison, MUMPS does not show good scalability, with an execution time of about 16 seconds, resulting in a real-time speed up factor of only 0.2, which is five times slower than real-time.

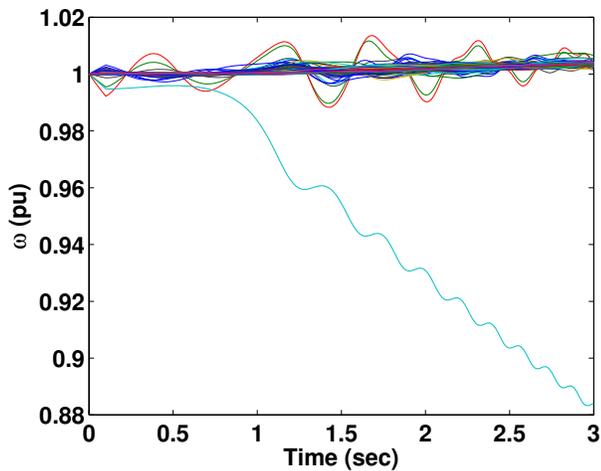


Fig. 6. Generator speeds for a three-phase fault applied for 0.1 seconds on bus 18

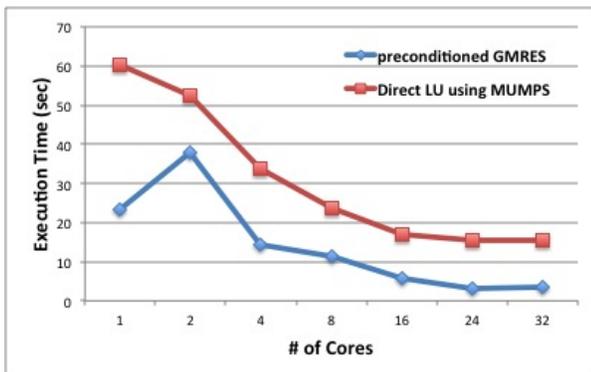


Fig. 7. Execution times for unstable 2383 bus system dynamics

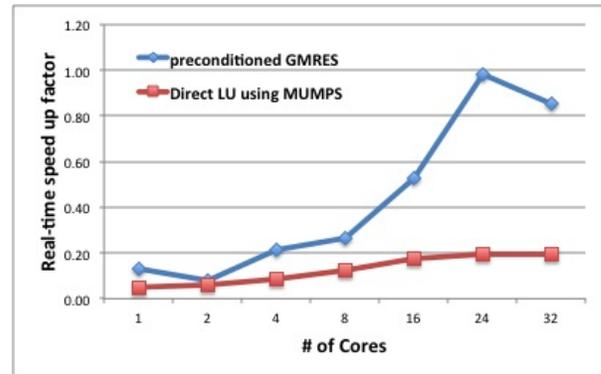


Fig. 8. Real-time speed up factor for unstable 2383 bus system dynamics

C. Comments on the scalability results

- For the single core case, the Block-Jacobi preconditioned GMRES scheme is faster than the direct solution using MUMPS. This can be attributed to (a) the difference in reordering scheme used (b) the direct solver algorithm. In MUMPS we experimented with different reordering schemes and selected one which had the least solution time. We could not get the information about the nonzeros in the factored matrix done by MUMPS. Moreover, PETSc's LU factorization data structure has recently been improved to make it more memory access efficient [26].
- Going from one core to two cores, an increase in the execution time is observed for the Newton-GMRES scheme. Note here that for two cores direct LU is only applied on the block diagonal, which results in a weaker preconditioner, as compared to LU on the entire matrix for the single core case. We observed an increase in the number of nonlinear iterations for the two core case as compared to single core case which explains the increase in the run-time.

V. PETSC: PORTABLE EXTENSIBLE TOOLKIT FOR SCIENTIFIC COMPUTATION

Developing scalable software for existing and emerging power system problems is a challenging task and requires considerable time and effort. This effort can be reduced by using high-performance software libraries that are tested on a wide variety of scientific applications, used on a gamut of platforms from single-core machines to supercomputers, have highly optimized implementations, and include a wide array of tested numerical solvers. Our simulator is based on the mathematical and computing framework of the high-performance library PETSc [8]. The wide range of sequential and parallel linear solvers, preconditioners, reordering strategies, flexible run-time options, ease of code implementation, debugging options, and a comprehensive source code profiler make PETSc an attractive experimentation platform for developing our parallel dynamics simulator. A review of PETSc and its use for developing scalable power system simulations can be found in [3].

PETSc is an open source package (BSD-style license) for the numerical solution of large-scale applications and provides

the building blocks for the implementation of large-scale application codes on parallel (and serial) computers. It is a part of Department of Energy's Advanced Computational Software[4] collection and was the winner of 2009 R&D Top 100 awards [24].

The PETSc package consists of a set of libraries for creating parallel vectors, matrices, and distributed arrays, scalable linear, nonlinear, and time-stepping solvers. The organization of PETSc is shown in Figure 9 and few of the components of each library are shown in Figure 10.

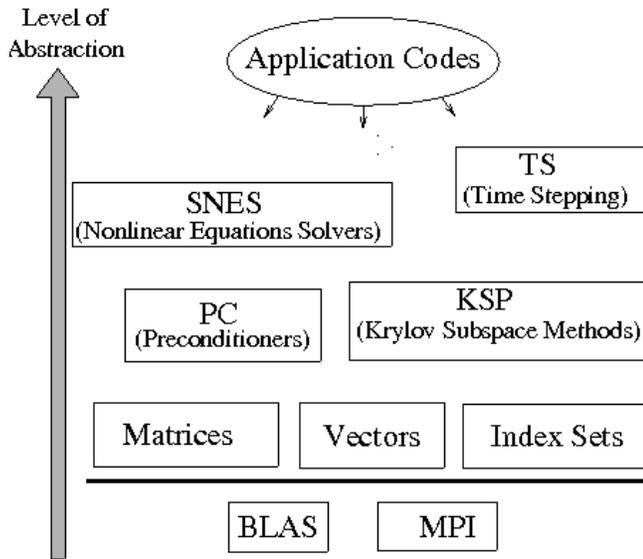


Fig. 9. Organization of the PETSc library [7]

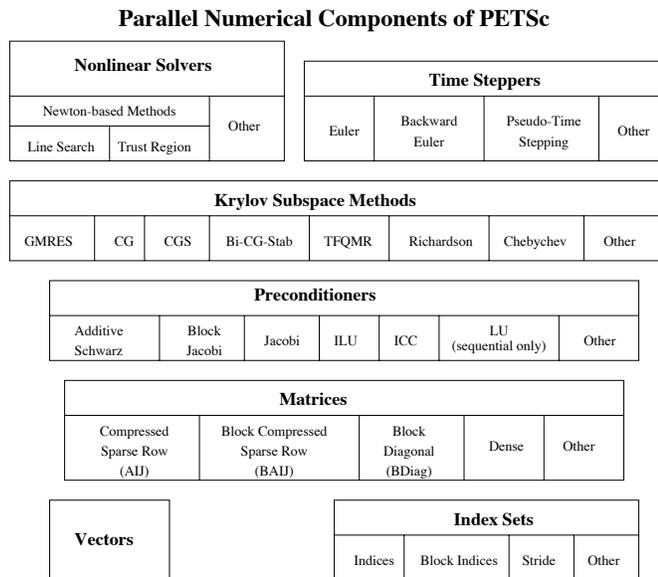


Fig. 10. Numerical libraries of PETSc [7]

To the authors knowledge, PETSc has not yet been used by the power system community for developing power system applications, but PETSc has been popular with researchers in the field of partial differential equation simulations.

VI. CONCLUSIONS

This paper presented real-time simulation of power system dynamics using a parallel Block-Jacobi preconditioned Newton-GMRES scheme. Results presented on a large 2383-bus system with 327 generators, for both stable and unstable operating conditions, show that real-time speed can be achieved with the proposed scheme. Nevertheless, the proposed scheme and other scalable algorithms need to be tested on various power system topologies and operating conditions, in order to evaluate their viability in an online environment. The PETSc library can accelerate the research on developing and testing various scalable algorithms for real-time dynamics simulation.

VII. ACKNOWLEDGEMENTS

This work was supported in part by the Office of Advanced Scientific Computing Research, Office of Science, U.S. Dept. of Energy, under Contract DE-AC02-06CH11357.

REFERENCES

- [1] Shirang Abhyankar. *Development of an implicitly coupled electromechanical and electromagnetic transients simulator for power systems*. PhD thesis, Illinois Institute of Technology, 2011.
- [2] Shirang Abhyankar, Alexander Flueck, Xu Zhang, and Hong Zhang. Development of a parallel three-phase transient stability simulator for power systems. In *Proceedings of the 1st International Workshop on High Performance Computing, Networking and Analytics for the Power Grid*. ACM, 2011.
- [3] Shirang Abhyankar, Barry Smith, Hong Zhang, and A. Flueck. Using PETSc to develop scalable applications for next-generation power grid. In *Proceedings of the 1st International Workshop on High Performance Computing, Networking and Analytics for the Power Grid*. ACM, 2011.
- [4] Advanced Computational Software (ACTS) Web page. <http://acts.nersc.gov>.
- [5] F. Alvarado. Parallel solution of transient problems by trapezoidal integration. *IEEE Transactions on Power Apparatus and Systems*, PAS-98:1080–1090, 1979.
- [6] P. R. Amestoy, I. S. Duff, J.-Y. L'Excellent, and J. Koster. A fully asynchronous multifrontal solver using distributed dynamic scheduling. *SIAM Journal on Matrix Analysis and Applications*, 23(1):15–41, 2001.
- [7] Satish Balay, Jed Brown, Kris Buschelman, Victor Eijkhout, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Barry F. Smith, and Hong Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 3.2, Argonne National Laboratory, 2011.
- [8] Satish Balay, Jed Brown, Kris Buschelman, Victor Eijkhout, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Barry F. Smith, and Hong Zhang. PETSc Web page. <http://www.mcs.anl.gov/petsc>, 2011.
- [9] J. Chai, S. Zhu, A. Bose, and D. J. Tylavsky. Parallel newton type methods for power system stability analysis using local and shared memory multiprocessors. *IEEE Transactions on Power Systems*, 6:9–15, 1991.
- [10] M. Crow and M. Ilic. The parallel implementation of waveform relaxation methods for transient stability simulations. *IEEE Transactions on Power Systems*, 5:922–932, 1990.
- [11] I. Decker, D. Falcao, and E. Kaszkurewicz. Parallel implementation of power system dynamic simulation methodology using the conjugate gradient method. *IEEE Transactions on Power Systems*, 7:458–465, 1992.
- [12] I. Decker, D. Falcao, and E. Kaszkurewicz. Conjugate gradient methods for power system dynamic simulation on parallel computers. *IEEE Transactions on Power Systems*, 7:629–636, 1994.
- [13] ERCOT. Electrical buses and outage scheduling, August 2006. <http://nodal.ercot.com>.
- [14] Joseph H. Eto and R. J. Thomas. Computational needs for the next generation electric grid. Technical report, U.S. Department of Energy, 2011. http://energy.gov/sites/prod/files/FINAL_CompNeeds_Proceedings2011.pdf.

- [15] D. Falcao. High performance computing in power system applications, September 1997. Invited Lecture at the 2nd International Meeting on Vector and Parallel Processing (VECPAR '96).
- [16] L. Hou and A. Bose. Implementation of the waveform relaxation algorithm on a shared memory computer for the transient stability problem. *IEEE Transactions on Power Systems*, 12:1053–1060, 1997.
- [17] Z. Huang and J. Nieplocha. Transforming power grid operations via high performance computing. In *Proceedings of the IEEE Power and Energy Society General Meeting 2008*, 2008.
- [18] M. Ilic, M. Crow, and M. Pai. Transient stability simulation by waveform relaxation methods. *IEEE Transactions on Power Systems*, 2:943–952, 1987.
- [19] Siemens PTI Inc. PSS/E Operation Program Manual: Volume 2, version 30.2.
- [20] Electric Power Research Institute. Grid transformation workshop results. Technical report, Electric Power Research Institute, 2012. http://my.epri.com/portal/server.pt?space=CommunityPage&cached=true&parentname=ObjMgr&parentid=2&control=SetCommunity&CommunityID=404&RaiseDocID=00000000001025087&RaiseDocType=Abstract_id.
- [21] PJM Interconnection. Pjm statistics, February 2011. <http://www.pjm.com>.
- [22] George Karypis and V. Kumar. ParMETIS: Parallel graph partitioning and sparse matrix ordering library. Technical Report 97-060, Department of Computer Science, University of Minnesota, 1997. <http://www.cs.umn.edu/metis>.
- [23] D. P. Koester, S. Ranka, and G. Fox. Power systems transient stability - a grand computing challenge. Technical report, School of Computer and Information Science and NorthEast Parallel Architectures Center (NPAC) Syracuse University, 1992. <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.54.7212>.
- [24] R&D Magazine. PETSc R&D 100 award, 2009. See <http://www.rdmag.com/Awards/RD-100-Awards/2009/07/PETSc-Release-3-0-Expands-Capabilities>.
- [25] Y. Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, 2000.
- [26] B. Smith and H. Zhang. Sparse triangular solves for ILU revisited: Data layout crucial to better performance. *International Journal of High Performance Computing Applications*, 25(4):386–391, 2011.
- [27] Ray Zimmerman and C. Murillo-Sanchez. Matpower 4.0 users manual. Technical report, Power Systems Engineering Research Center, 2011.



Shrirang Abhyankar received the M.S. degree (2006) and the Ph.D. degree (2011) in electrical engineering from Illinois Institute of Technology, Chicago. He is currently a post-doctoral appointee in the Mathematics and Computer Science Division at Argonne National Laboratory. His research interests include scalable algorithms for large-scale transient stability analysis, and combined electromechanical and electromagnetic simulation, multi-physics and multi-scale algorithms, and application development in the hybrid MPI-shared memory environment.



Alexander J. Flueck received the B.S. degree (1991), the M.Eng. degree (1992) and the Ph.D. degree (1996) in electrical engineering from Cornell University. He is currently an associate professor at Illinois Institute of Technology in Chicago. His research interests include three-phase transient stability, electromagnetic transient simulation, autonomous agent applications to power systems, transfer capability of large-scale electric power systems, contingency screening with respect to voltage collapse, and parallel simulation of power systems

via message-passing techniques on distributed-memory computer clusters.

Portions of the submitted manuscript have been created by UChicago Argonne, LLC, Operator of Argonne National Laboratory (“Argonne”). Argonne, a U.S. Department of Energy Office of Science laboratory, is operated under Contract No. DE-AC02-06CH11357. The U.S. Government retains for itself, and others acting on its behalf, a paid-up nonexclusive, irrevocable worldwide license in said article to reproduce, prepare derivative works, distribute copies to the public, and perform publicly and display publicly, by or on behalf of the Government.