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 due to the need of solving 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 the Newton's method. We present a parallel linear solution scheme using Krylov subspace based iterative solver GMRES with a Block-Jacobi preconditioner that shows promising prospect of a real-time dynamics simulation. The proposed linear solution scheme shows 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 one of the primary focus of the power system community in recent years. This view was reiterated in the recent DOE and EPRI workshops [13], [19]. More than two decades ago, real-time dynamics simulation was identified as a 'grand computing challenge' [22] and that view exists to date. 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.

Dynamics simulation of a large-scale power system is computationally challenging due to 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 differential-algebraic equations, where f and g are vector-valued nonlinear functions.

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

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

- A numerical integration scheme to convert the differential equations in algebraic form,
- A nonlinear solution scheme to solve the resultant nonlinear algebraic equations, and
- 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. As system size increases, execution times grow dramatically. Thus 'real- time' dynamics analysis of a utility or a regional operator network is an enormous computing challenge.

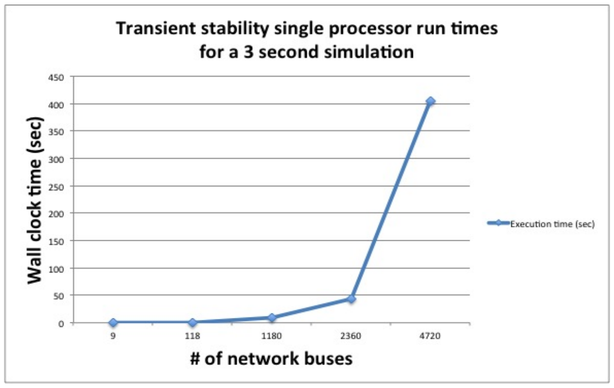

Fig. 1. Dynamic simulation execution times for different systems on a single processor

For example, PJM, a regional transmission organization (RTO) covering 168,500 square miles of 12 different states, monitors approximately 13,500 buses [20]. Similarly, the Electric Reliability Council of Texas (ERCOT) monitors approximately 18,000 buses [12]. 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 off-line. 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 [16].

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 USA.

II. Speeding power grid dynamics simulation via parallel computing

A natural way to speed up this computation is to use parallel computing techniques, i.e., share the computational load amongst multiple processors. The need for parallelizing existing power system applications is even greater as the computer hardware industry moves towards 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 via parallel algorithms that distribute the computational load over multiple cores. Several workshops [13], [19] have highlighted the need for investigating these multicore/manycore architectures for accelerating performance of power system applications.

In the context of parallel algorithms for dynamic simulations, most of the research effort was done over the last 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, i.e., to reduce the inter-processor communication, and balance the work load. Once a suitable partitioning strategy is selected the next key thing is the solution of the linear system in each Newton iteration. Several linear solution schemes have proposed in the literature, of which the prominent are the *Very Dishonest Newton Method* and *Conjugate gradient*. Reference [8] 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 [10] decomposes the network equations in a Block Bordered Diagonal Form (BBDF) and then uses a hybrid solution scheme using LU and Conjugate gradient. Reference [11] solves the network by a block-parallel version of the preconditioned conjugate gradient method. The network matrix in [11] is in the Near Block Diagonal Form (NBDF).

The Parallel-in-time approach was first introduced in [5]. The idea of this approach was 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 [17], [9], [15] 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 [14]. 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 (multi-rate integration).

III. Parallel dynamics simulation approach

This section describes the details of the parallel implementation of our developed dynamics simulator. The dynamics simulator is a three-phase dynamics simulator, unlike the existing dynamics simulator that use a per-phase balanced network model. The details of the developed three-phase dynamics simulator can be found in [1], [2].

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 2 subnetworks. Each subnetwork is then

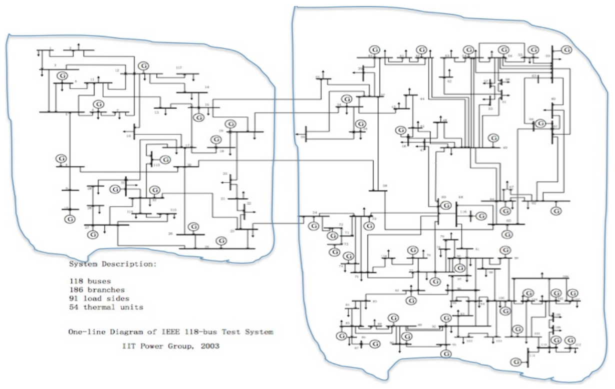

Fig. 2. Partitioning of the IEEE 118 bus system for 2 processors

the domain of operation of a processor, i.e., 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, while subscript c represents the variables needed from other processors to compute the function on the current processor.

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

Note that differential equations, i.e. the electromechanical machine equations, are naturally decoupled as they are only incident at a bus while the algebraic network equations require communication with other processors to compute their local function. Hence the partitioning of the network is done only using 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 developed simulator, we use the ParMetis package [21] 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 (a) minimizing the edge cuts while (b) having balanced partitions, i.e., 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 also can be assigned to each vertex to account for different computational requirement. With vertex weights, ParMetis tries to minimize the edge cuts and also have the same amount of vertex weights in each sub-domain.

For our developed 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

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

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

$$Ax = b \tag{4}$$

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

$$b - Ax_m \perp L_m$$

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

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

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)[24] 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 mth krylov subspace

$$||b - Ax_m||_2 = min||b - Ax||_2 \tag{5}$$

At step m, an arnoldi process is applied for the mth krylov subspace to generate the next basis vector. When the norm of the new basis vector is sufficiently small, GMRES solves the minimization problem

$$y_m = argmin||\beta e_1 - \bar{H_m}y||_2$$

where $\bar{H_m}$ is the $(m+1)xm$ 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 which 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

$$M^{-1}Ax = M^{-1}b \tag{6}$$

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

$$AM^{-1}y = b, \qquad x = M^{-1}y \tag{7}$$

or split preconditioning

$$M_1^{-1}AM_2^{-1}y = M_1^{-1}b, \qquad x = M^{-1}y \tag{8}$$

where the preconditioner is $M = M_1M2$.

When Krylov subspace methods are used, it is not necessary to form the preconditioned matrices $M^{-1}A$ or AM^{-1} explicitly since this would be too expensive. 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 this 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

$$\begin{array}{c} [0] \\ [1] \end{array} \left[\begin{array}{cc} J_1 & J_2 \\ \hline J_3 & J_4 \end{array}\right]$$

then the parallel Block-Jacobi preconditioner is

$$\begin{array}{c} [0] \\ [1] \end{array} \left[\begin{array}{cc} J_1^{-1} & \\ \hline & J_4^{-1} \end{array}\right]$$

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 PETSc, on the test systems to determine the optimal reordering strategy, i.e., the ordering scheme resulting in the least number of nonzeros in the factored matrix. The *Quotient Minimum Degree* ordering was found to be the most optimal approach for the systems that we tested.

IV. Scalability Results

The 2383 bus system provided with the MatPower [26] 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 2455 MW. This case represents the Polish 400, 220 and 110 kV networks during winter 1999-2000 peak conditions. It is part of the 7500+ bus European UCTE system. For the dynamic simulations, all the generators were modeled using the GENROU model with an IEEE Type 1 exciter model [18] with the loads modeled as constant impedance loads. The numerical integration scheme used is an implicit trapezoidal scheme with a time step of 1 cycle or 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) to assess the proximity of the simulation to real-time. A value of $s_r \geq 1$ indicates that the simulation is running in real-time or faster than real-time.

$$s_r = \frac{T_s}{T_e} \tag{9}$$

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 preconditioned GMRES scheme and compare it with parallel sparse LU factorization using the MUMPS [6] package. Two cases are considered to assess the scalability (i) A three-phase fault on bus 185 for 6 cycles that results in stable dynamics, and (ii) A three-phase fault on bus 18 for 6 cycles that results in unstable dynamics. The dynamics of the 2383 bus system were simulated for a period of 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 6 cycles from t=0.0 sec to t=0.01 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.

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 speed up as compared to a parallel LU factorization using MUMPS.

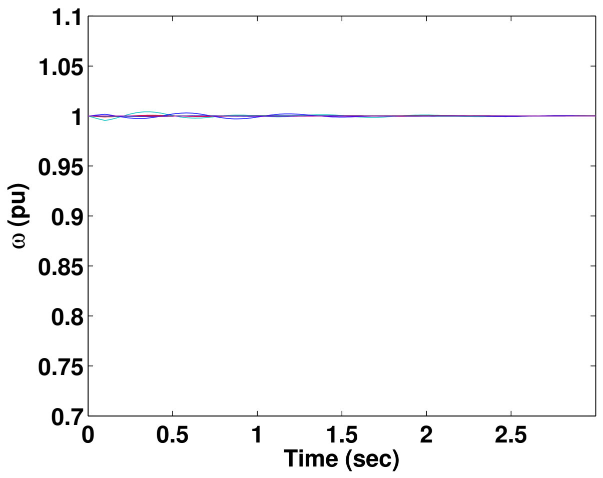

Fig. 3. Generator speeds for a three-phase fault applied for 6 cycles on bus 185

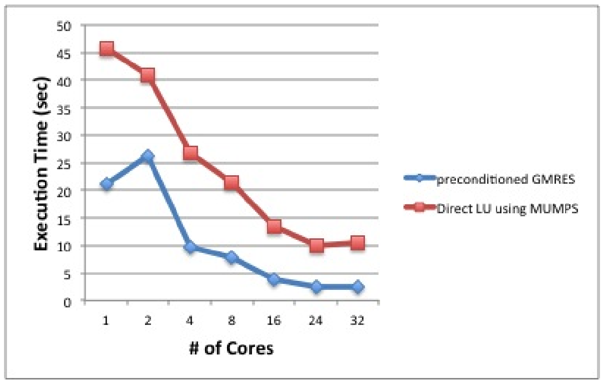

Fig. 4. Execution times for stable 2383 bus system dynamics

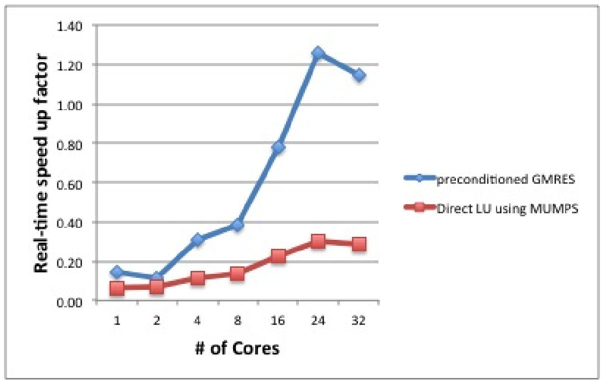

Fig. 5. Real-time speed factor for stable 2383 bus system dynamics

B. Unstable case

For testing the unstable dynamics, a three-phase fault was placed on bus 18 for six cycles from t = 0.0 sec to t = 0.1 sec. Bus 18 has a generator incident on it having the largest power dispatch. As seen from 6 following the fault the generator on bus 18 loses synchronism and its speed drops. The unstable dynamics result in increased number of nonlinear, and linear, iterations due to to increased swinging, or seperation, of the generator speeds. For the stable case, the maximum number of Newton iterations taken by the solver was 3 while for the unstable case it was 5. As seen in figures 7, the execution time on 1 processor using the Block-Jacobi preconditioned GMRES scheme takes about 22 seconds of execution time while on 24 cores it takes around 3 seconds, i.e. equal to the simulation time length. Thus, real-time simulation is 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 factor of only 0.2.

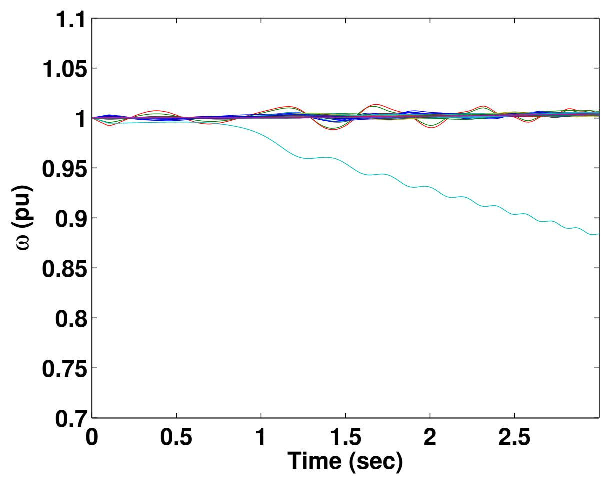

Fig. 6. Generator speeds for a three-phase fault applied for 6 cycles on bus 18

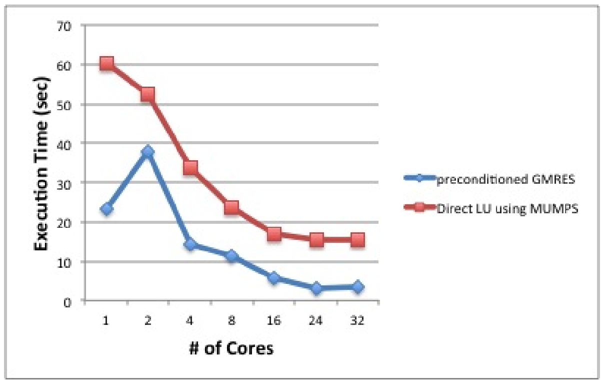

Fig. 7. Execution times for unstable 2383 bus system dynamics

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, such as PETSc, which are tested on a gamut of scientific applications, used on single-core machines to supercomputers, have highly optimized implementations, and a wide array of tested numerical solvers. High performance libraries have not yet been used by the power system community for developing power system applications, but such libraries have been well explored by researchers doing PDE simulations.

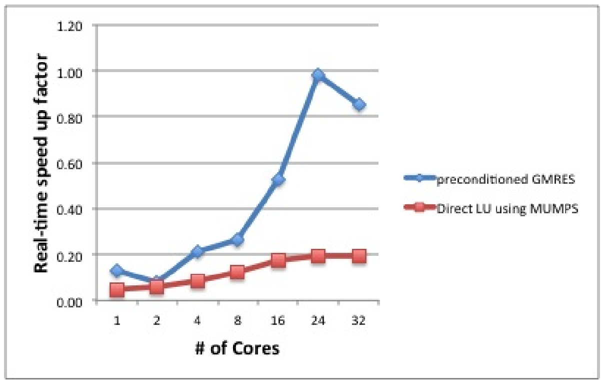

Fig. 8. Real-time speed factor for unstable 2383 bus system dynamics

The Portable, Extensible Toolkit for Scientific Computation (PETSc)[7] 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 [23]. PETSc has been also cited as one of the Top 10 advances in computational science accomplishments of the U.S. Department of Energy in 2008 [25]. PETSc is funded primarily by the United States Department of Energy, Office of Science, by the Advanced Scientific Computing Research's (ASCR) Applied Mathematics Research and Scientific Discovery through Advanced Computing (SciDAC) programs.

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.

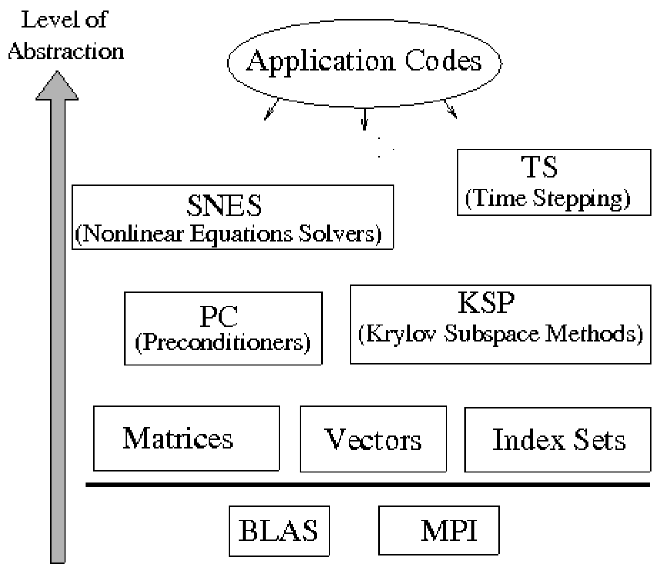

Fig. 9. Organization of the PETSc library [7]

It is an open source package for the numerical solution of large-scale applications and is free for anyone to use (BSD-style license). It runs on operating systems such as Linux, Microsoft Windows, Apple Macintosh, and Unix. It can be used from within the Microsoft Developers Studio. PETSc can be configured to work with real or complex data types (not mixed though), single or double precision, and 32 or 64-bit integers. It has been tested on a variety of tightly coupled parallel architectures such as Cray XT/5, Blue Gene/P, and Earth Simulator, and also on loosely coupled architectures such as networks of workstations.

PETSc uses a plug-in philosophy to interface with external softwares. Various external software packages such as SuperLU, SuperLU_Dist, ParMetis, MUMPS, PLAPACK, Chaco, and Hypre can be installed with PETSc. PETSc provides an interface for these external packages so that they can be used in PETSc application codes.

Allowing the user to modify parameters and options easily at runtime is very desirable for many applications. For example, the user can change the linear solution scheme from GMRES to direct LU factorization, or can change the matrix storage type, or preconditioners, via run-time options.

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 good review of PETSc, and its use for developing scalable power system simulations, can be found in [3].

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 via the proposed scheme. Although the proposed scheme shows that real-time power system dynamics simulation is achievable, the proposed scheme, and other scalable algorithms, need to be tested on various power system topologies and operating conditions to test 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.

References

[1] Shrirang Abhyankar. *Development of an implicitly coupled electromechanical and electromagnetic transients simulator for power systems*. PhD thesis, Illinois Institute of Technology, 2011.

[2] Shrirang 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] Shrirang 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 dynami c 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] 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.

[9] 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.

[10] 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.

[11] 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.

[12] ERCOT. Electrical buses and outage scheduling, August 2006. http://nodal.ercot.com.

[13] 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.

[14] 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).

[15] 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.

[16] 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.

[17] M. Ilic, M. Crow, and M. Pai. Transient stability simulation by waveform relaxation methods. *IEEE Transactions on Power Systems*, 2:943–952, 1987.

[18] Siemens PTI Inc. PSS/E Operation Program Manual: Volume 2, version 30.2.

[19] 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=000000000001025087&RaiseDocType=Abstract_id.

[20] PJM Interconnection. Pjm statistics, February 2011. http://www.pjm.com.

[21] George Karypis and V. Kumar. ParMETIS: Parallel graph partitioning and sparse matrix o rdering library. Technical Report 97-060, Department of Computer Science, University of Minneso ta, 1997. http://www.cs.umn.edu/ metis.

[22] 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.

[23] 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.

[24] Y. Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, 2000.

[25] B. Smith et al. Prizes Won Using PETSc. http://www.mcs.anl.gov/petsc/publications/prizes.html.

[26] Ray Zimmerman and C. Murillo-Sanchez. Matpower 4.0 users manual. Technical report, Power Systems Engineering Research Center, 2011.

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.

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 combined electromechanical and electromagnetic simulation, parallel algorithms for large-scale dynamic analysis of power systems.

Alexander J. Flueck 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.