

Using PETSc to Develop Scalable Applications for Next-Generation Power Grid

Shrirang Abhyankar
Department of Electrical and
Computer Engineering
Illinois Institute of Technology
3301 South Wabash Avenue
Chicago, Illinois - 60616
abhyshr@iit.edu

Barry Smith
Mathematics and Computer
Science Division
Argonne National Laboratory
9700 South Cass Avenue
Argonne, Illinois
bsmith@mcs.anl.gov

Hong Zhang
Mathematics and Computer
Science Division
Argonne National Laboratory
9700 South Cass Avenue
Argonne, Illinois
hzhang@mcs.anl.gov

Alexander Flueck
Department of Electrical and
Computer Engineering
Illinois Institute of Technology
3301 South Wabash Avenue
Chicago, Illinois - 60616
flueck@iit.edu

ABSTRACT

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. This paper introduces the high performance library PETSc and motivates using such high performance libraries for developing existing and future power system applications.

Categories and Subject Descriptors

G.4 [Mathematical Software]: Parallel and vector implementations

Keywords

Power System Applications, High Performance Computing Library, PETSc, Parallel implementation

1. INTRODUCTION

The electricity industry is growing through a revolution of new technologies and ideas to make the existing grid more secure, reliable and interconnected. The penetration of wind, solar, and other renewable resources of electricity production is increasing. The advent of deregulation is driving the

power industry toward economic operation and thus operating the transmission system to its fullest potential. Smart Grid is bringing a new meaning to how communication and control are done. The incorporation of power electronics equipment in power systems is increasing and brings with it non-fundamental frequency harmonics. In order to manage the load growth, and to enhance reliability and security, the interconnection between utility controlled transmission systems is growing. As these interconnections continue to grow, there will be a need for managing large-scale and ultra-large-scale transmission systems, whether regional, national, or multi-national, in real time.

These developments are making power system computational problems more challenging in terms of modeling complexity and faster simulation requirements. The ultimate goal for power system applications is to have high-fidelity models along with real-time processing speed to provide look-ahead or proactive decision making. Thus, the use of parallel machines to speed the applications is critical for future applications.

Considerable research into parallel algorithms for power system applications has been done. A literature review shows that power system researchers have developed parallel algorithms for applications such as transient stability simulation, power flow, state estimation, optimal power flow, electromagnetic transients simulations, and contingency analysis. The most dominant research effort has been in transient stability applications to solve the resultant differential-algebraic equations, or DAEs, for the power system.

However, development of parallel algorithms requires considerable time and effort through various phases of application development, including partitioning, managing communication between processors, nonlinear function and Jacobian evaluation and debugging, with most time spent on implementing the linear solver. Moreover, various parallel algorithms must be benchmarked on different system topologies to select the optimal (or set of optimal algorithms). Benchmarking with various solvers is a task that entails considerable time and effort. As a result, parallel power system

applications are developed using by a specific linear solution scheme tested on a specific architecture for a given power system topology.

In this paper, we present the high performance computing library PETSc, developed at Argonne National Laboratory, that can aid in reducing this experimentation time. The range of linear solvers and preconditioners, abstract linear algebra object interfaces for writing user application codes, portability to different operating systems, and flexible run-time options make PETSc an attractive platform for developing scalable power system applications.

2. PETSC: PORTABLE EXTENSIBLE TOOLKIT FOR SCIENTIFIC COMPUTATION

The Portable, Extensible Toolkit for Scientific Computation (PETSc) is a suite of data structures and routines that provide the building blocks for the implementation of large-scale application codes on parallel (and serial) computers. PETSc uses the MPI standard for all message-passing communication. PETSc includes an expanding suite of parallel linear, nonlinear equation solvers and time integrators that may be used in application codes written in Fortran, C, C++, Python, and MATLAB (sequential). PETSc provides many of the mechanisms needed within parallel application codes, such as parallel matrix and vector assembly routines. The library is organized hierarchically, enabling users to employ the level of abstraction that is most appropriate for a particular problem. By using object-oriented programming, PETSc provides enormous flexibility for users.

PETSc consists of a variety of libraries (similar to classes in C++), see 1. Each library manipulates a particular family of objects (for instance vectors) and the operations one would like to perform on the objects. The objects and operations in PETSc are derived from our long experience with scientific computation.

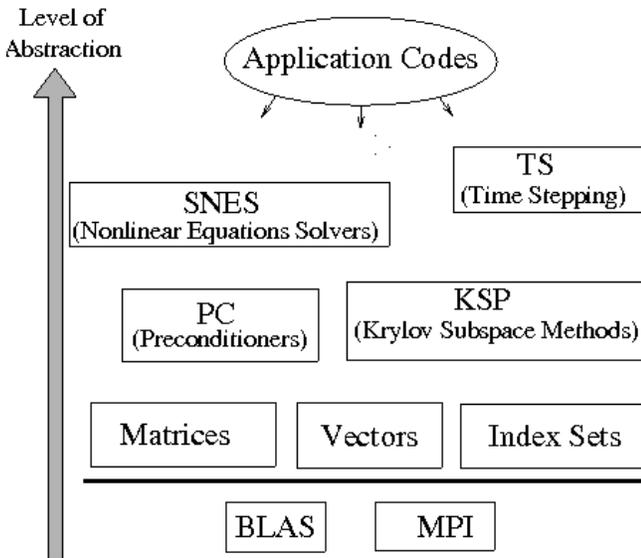


Figure 1: Organization of the PETSc library [1]

Each object consists of an abstract interface (simply a set of calling sequences) and one or more implementations using

particular data structures. Thus, PETSc provides clean and effective codes for the various phases of solving applications, with a uniform approach for each class of problems. This design enables easy comparison and use of different algorithms (for example, to experiment with different Krylov subspace methods, preconditioners, or truncated Newton methods). The libraries enable easy customization and extension of both algorithms and implementations. This approach promotes code reuse and flexibility, and separates the issues of parallelism from the choice of algorithms. Hence, PETSc provides a rich environment for modeling large-scale scientific applications as well as for rapid algorithm design and prototyping.

Parallel Numerical Components of PETSc

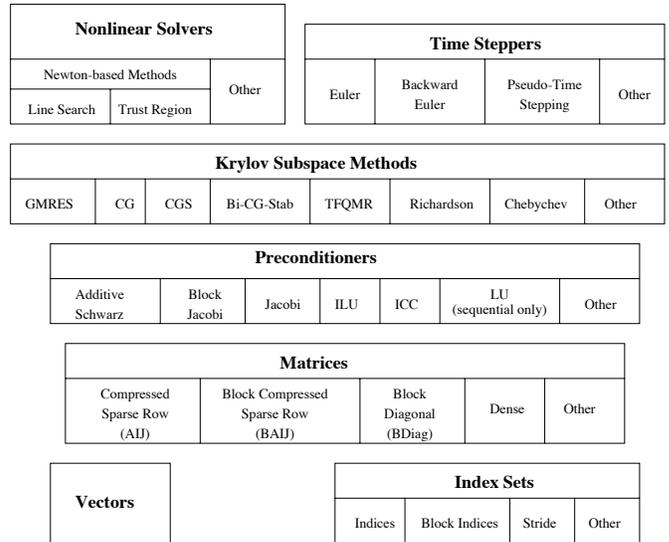


Figure 2: Numerical libraries of PETSc [1]

3. PETSC FEATURES

PETSc 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 Hyprc 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.

If an application uses a large number of parameters, these can be also supplied by a text file that is read when the PETSc code begins.

Debugging is one of the most pain staking task in application code development. PETSc provides various features to ease the debugging process. Debuggers such as gdb, dbx, xxgdb, etc., can be used, debugger can be either activated at the start of the program or when an error is encountered. Moreover, a subset of processes can be selected for debugging parallel application codes. In addition, the widely used package Valgrind can be used for detecting memory errors. Jacobian computation for the solution of nonlinear system via Newton’s method is cumbersome and a great deal of time and effort can be spent in debugging the Jacobian. PETSc provides run-time options to check the user’s Jacobian entries by comparing them with a finite-difference approximated Jacobian.

As PETSc developers, we are actively involved in responding user queries, and PETSc development has benefited tremendously through these queries. A comprehensive manual is available on the PETSc website, and each library in PETSc has many examples demonstrating how to use that library.

PETSc automatically logs object creation, times, and floating-point counts for the library routines. Users can easily supplement this information by monitoring their application codes. The users can either log their routines, called *event* logging, or log multiple sections of the code, called *stage* logging.

4. PETSC LIBRARIES

This section describes the various libraries, or classes, available with the PETSc library.

4.1 Vectors

The vector (denoted by Vec) is one of the simplest PETSc objects. Vectors are used to store solutions, right-hand sides for linear systems, and so forth. PETSc currently provides several basic vector types; the two most commonly used are sequential and parallel (MPI-based). Basic vector operations, such as dot product and sum, are available in the PETSc vector library. The comprehensive list of vector operations can be found in [3]. Assigning values to individual components of the parallel vector can be done either using global numbering or using a local process numbering. With global numbering, any process can set any components of the vector; PETSc ensures that they are automatically stored in the correct location. PETSc vectors have full support for general scatters and gathers. One can select any subset of the components of a vector to insert or add to any subset of the components of another vector. We refer to these operations as generalized scatters, though they are actually a combination of scatters and gathers.

4.2 Matrices

PETSc provides a variety of matrix implementations because no single matrix format is appropriate for all problems. Currently we support dense storage and compressed sparse row storage, as well as several specialized formats such as blocked compressed sparse row, and symmetric compressed and block compressed formats. All the matrices are available as sequential and parallel versions. An interface for adding user-defined matrix formats is also provided. Most power system applications use a skyline storage format for

the matrices that can be easily added and used with the linear solver.

4.3 Linear solvers and preconditioners

The object KSP is the heart of PETSc: it provides uniform and efficient access to all of the package’s linear system solvers, including parallel and sequential, direct and iterative. KSP is intended for solving nonsingular systems of the form

$$Ax = b \tag{1}$$

where A denotes the matrix representation of a linear operations, b is the right hand side vector, and x is the solution vector. KSP uses the same calling sequence for both direct and iterative solution of a linear system. In addition, particular solution techniques and their associated options can be selected at runtime.

The combination of a Krylov subspace method and a preconditioner is at the center of most modern numerical codes for the iterative solution of linear systems. Since the rate of convergence of Krylov projection methods for a particular linear system is strongly dependent on its spectrum, preconditioning is typically used to alter the spectrum and hence accelerate the convergence rate of iterative techniques. Preconditioning can be applied to the system in (1) by

$$(M_L^{-1}AM_R^{-1})(M_Rx) = M_L^{-1}b \tag{2}$$

where M_L and M_R indicate left and right preconditioning matrices. By default, all KSP implementations use left preconditioning. Right preconditioning can be activated for some methods by a run-time option or by calling a routine. Currently, PETSc supports over 20 KSP methods and preconditioners. A partial list of the available preconditioners is given in Table 1. The preconditioner type PCComposite al-

Table 1: Partial list of PETSc preconditioners

Preconditioners
Jacobi
Block Jacobi
SOR (and SSOR)
Incomplete Cholesky
Incomplete LU
Additive Schwartz
Combination of preconditioners
LU
Cholesky
Shell for user-defined preconditioner

lows one to form new preconditioners by combining already defined preconditioners solvers. Solvers and preconditioners can be also nested. For example, with a parallel Block-Jacobi preconditioner, (i.e., the preconditioner formed using the diagonal block of the matrix on each processor), any of the other preconditioners, such as LU, ILU, or SOR, can be used on the block.

Various reordering schemes to reduce the fill-in for the factored matrices are also available and can be accessed either by calling routines or by run-time options. The current

reordering schemes in PETSc are given in Table 2. User-defined reordering schemes can be easily included too.

Table 2: Partial list of reordering schemes

Reordering schemes
Natural
Nested dissection
Reverse Cuthill-McKee
1-way dissection
Quotient minimum degree
Row length

4.4 Nonlinear solvers

The nonlinear solver class SNES includes methods for solving systems of nonlinear equations of the form

$$F(x) = 0 \tag{3}$$

Newton-like methods provide the core of the package, including both line search and trust region techniques. Built on top of the linear solvers and data structures discussed in preceding sections, SNES enables the user to easily customize the nonlinear solvers according to the application at hand. Also, the SNES interface is identical for the uniprocess and parallel cases; the difference in the parallel version is that each process typically forms only its local contribution to various matrices and vectors. To access the SNES solver, the user provides a C, C++, Fortran, Python, or MATLAB routines to evaluate the nonlinear function in Equation (3) and Jacobian. PETSc also provides routines to approximate the Jacobian by finite differences if an analytical expression for the Jacobian is not available or if is too hard to compute.

4.5 Time-stepping integrators

The TS library in PETSc provides a framework for the scalable solution of ODEs and DAEs and of steady-state problems using pseudo-time stepping. Various numerical integration algorithms such as explicit and implicit Euler schemes, implicit trapezoidal integration and multi-stage explicit Runge-Kutta variable time stepping schemes are available in PETSc. TS uses the SNES and the KSP objects to solve the underlying nonlinear/linear system and the user can tune this solution process at run time too.

5. NEW ADDITIONS TO THE PETSC LIBRARY

This section presents the recent additions to the PETSc library some of which were already in the previously released version and some of which will be incorporated in the next release version.

5.1 Memory efficient data structure for LU factorization

A memory efficient data structure for LU and incomplete LU factorization [13], for sparse compressed row as well as blocked matrix formats, was added in the previous release version of PETSc. This newly developed data structure stores the L and U matrices separately and stores the entries in U in a reverse direction (i.e., starting from the last row to

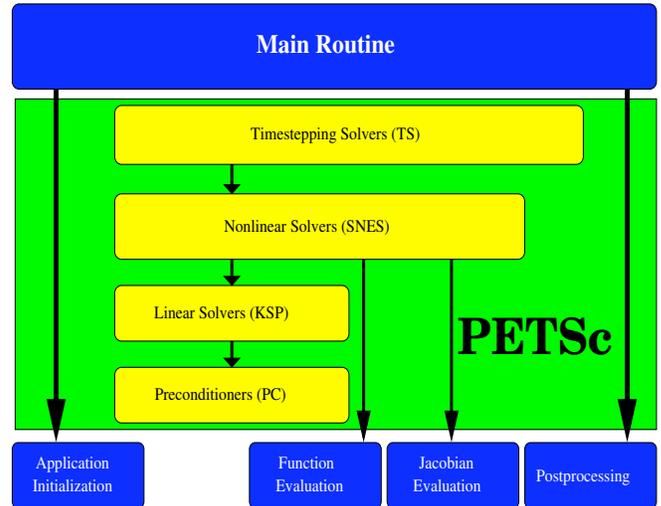


Figure 3: Flow control of PETSc application [1]

the first). This reorganization of the data structure results in a continuous access of LU elements and thus provides better memory access.

5.2 Support for GPGPU

The most recently released version of PETSc provides support for solving applications on NVidia GPUs. PETSc uses the CUSP and Thrust libraries, developed by NVidia, to do computations on the GPU. Essentially, the user writes code using the PETSc libraries, and PETSc solves the application on the GPU and handles the communication of the data to and from the GPU to the CPU.

5.3 PETSc-MATLAB interface

MATLAB is a popular prototyping language for rapid code development. We have developed an interface for using the PETSc libraries from MATLAB for sequential computation. Almost all the current PETSc libraries and operations are available through the PETSc-MATLAB interface.

5.4 Support for hybrid MPI-shared memory

The most recent release of PETSc contains a vector and matrix class that use POSIX pthreads for computation within a multicore/multichip shared-memory node.

5.5 Multiphysics preconditioners

The simulation of multiphysics and multiscale models is a challenging topic in the field of numerical computation. PETSc provides an efficient multiphysics preconditioner class, called *fieldsplit*, for preconditioning coupled multiphysics problems. This preconditioner allows the use of a custom linear solver for each physics domain, along with its own preconditioner, reordering strategy, and all the other intricacies. Four *fieldsplit* preconditioners are available. For example, if the linear system to be solved comprises two physics and described by

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix} \tag{4}$$

the available *fieldsplit* preconditioners are

- Block-Jacobi or additive

$$\begin{bmatrix} A^{-1} & \\ & D^{-1} \end{bmatrix} \quad (5)$$

- Block-Gauss-Siedel or multiplicative

$$\begin{bmatrix} A & \\ C & D \end{bmatrix}^{-1} \quad (6)$$

- Symmetric Block Gauss-Siedel

$$\begin{bmatrix} A & \\ & 1 \end{bmatrix}^{-1} \left(1 - \begin{bmatrix} A & B \\ & 1 \end{bmatrix} \begin{bmatrix} A & \\ C & D \end{bmatrix}^{-1} \right) \quad (7)$$

- Schur-complement based

$$\begin{bmatrix} 1 & \\ CA^{-1} & 1 \end{bmatrix} \begin{bmatrix} A & B \\ & S \end{bmatrix}, \quad (8)$$

where

$$S = D - CA^{-1}B$$

6. USING PETSC AND SUPPORTING LIBRARIES FOR DEVELOPING POWER SYSTEM APPLICATIONS

This section describes a few applications that can be developed by using the PETSc and two other libraries, TAO and SLEPc, developed using PETSc.

6.1 Combined electromechanical electromagnetic transients simulation

The number of power electronics devices is expected to increase for a more flexible control of power systems. Therefore, the effect of non-fundamental frequency harmonics will increase, and the dynamic simulation will entail modeling of power electronics devices via an electromagnetic simulation. An attractive way of incorporating the simulation of non-fundamental frequency harmonics in an electromechanical transient simulator is via a hybrid simulation[21]. In a hybrid simulator, most of the bulk power system is modeled by using an electromechanical transients simulator, while a small part of it is modeled by using electromagnetic transients simulator. The PETSc library provides efficient data structures to ease the development of multiscale or multiphysics applications and is an attractive platform for the development of a combined electromechanical-electromagnetic transients simulation. The development of a parallel implicitly coupled electromechanical and electromagnetic transients simulator, using the PETSc library, is detailed in [7]. The authors in [7] have reported a speed up of about 4.6 times on six cores for the developed parallel implicitly coupled simulator using PETSc for a large-scale power system.

6.2 Combined transmission-distribution analysis

Various ISOs have indicated the need to model and gather real-time information from the sub-transmission and distribution systems in order to enhance reliability and awareness [9], in other words, to provide finer granularity modeling. While ISOs have traditionally been able to forecast load within a 2% error, deployment of distributed energy resources and utility-scale storage may increase the error

substantially [9]. Moreover, new demands are being placed on the power infrastructure as a result of the introduction of plug-in vehicles. The sheer volume of the components for a combined transmission-distribution analysis presents an onerous computational task and emphasizes the need for developing parallel algorithms for such an analysis and the use of high performance computing libraries.

6.3 Electromechanical transients simulation

Considerable research on parallel implementation of transient stability application has been done by power system researchers [14]-[20], as it offers a possibility of real-time dynamic simulation. The transient stability formulation is defined by the following differential algebraic model of the system:

$$\begin{aligned} \dot{x} &= f(x, y) \\ 0 &= g(x, y) \end{aligned} \quad (9)$$

The TS library in PETSc can be used for solving nonlinear differential algebraic transient stability equations in (9). The TS library in PETSc provides various DAE solvers including the implicit-trapezoidal integration scheme. The recent release version of PETSc has added Implicit-Explicit (IMEX) time integration schemes for multirate problems which could be also experimented with.

A Schur-complement-based linear scheme is generally preferred for the solution of the nonlinear algebraic system obtained by discretization of (9). This scheme can be selected at run-time via PETSc's multiphysics schur-complement based solver by simply specifying sets of indices for the generator and the network blocks. Various conjugate-gradient-based algorithms (conjugate gradient, conjugate gradient square, bi-conjugate gradient) are also available and can be selected at run time.

Reference [7] details the development of a parallel three-phase transient stability simulator built by using the PETSc library and presents the scalability results using several linear solution strategies. The speedup for three large scale systems using the iterative solver GMRES with a very dishonest Block-Jacobi preconditioner is shown in Figure 4.

6.4 Power Flow related applications

Power flow is a fundamental application in power system analysis. Various analyses such as steady state security, area power transfer studies, contingency screening, require a power flow solution. Essentially, the power flow problem solves the nonlinear power balance equation for the network given a generation set point and a load injection,

$$F(x) = 0 \quad (10)$$

where x are the bus voltages. In addition, a power flow solution serves as a starting point for dynamic simulations or short circuit studies. Faster solution of power flow equations would speed power system analysis programs entailing repeated power flow solutions such as contingency analysis, continuation power flow.

The nonlinear solver class SNES can be used for developing parallel power flow applications where the user needs to only provide a routine for the nonlinear function evaluation (and an optional Jacobian evaluation). The underlying linear solver can be selected at run time; for example, the linear solver can be switched from GMRES without a preconditioner to direct factorization at run time.

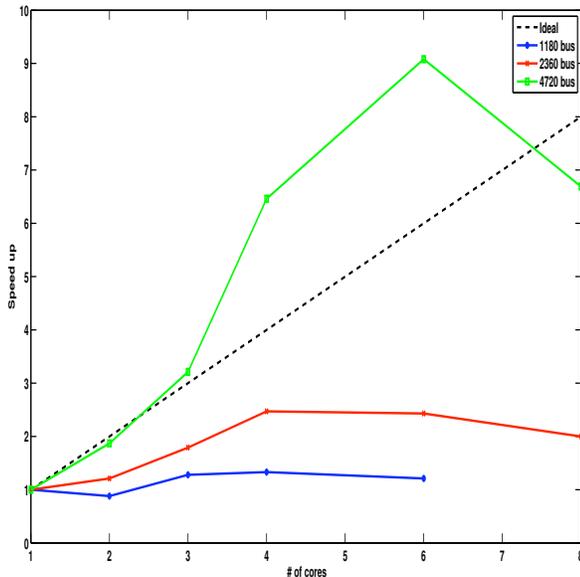


Figure 4: Speedup of parallel three-phase transient stability simulator

6.5 Power system optimization

PETSc has been developed primarily for solving linear and nonlinear algebraic equations and as such has limited support for optimization applications. The scalable optimization library TAO (Toolkit for Applied Optimization) [5], which is built using PETSc can be used for applications such as optimal power flow, security constrained optimal power flow. Support for solving security constrained unit commitment applications needing mixed integer linear/nonlinear solvers has not yet been developed in PETSc or TAO.

6.6 Small Signal Stability

SLEPc (Scalable library for Eigenvalue Problem Computations) [6], a library based on PETSc, can be used for developing small signal stability analysis applications. SLEPc consists of several algorithms for scalable computation of eigenvalues and uses various PETSc data structures such as matrix storage schemes, and vectors.

6.7 Electromagnetic Transients Simulation

The ultimate goal for the power system simulation researchers is an electromagnetic transient simulation in real-time. However, the modeling complexity along with time-step limitations are overwhelming. The transmission lines for electromagnetic transients simulation are modeled by traveling wave equations consisting of two disjoint equivalent circuits. This model is nicely structured for parallel processing with equations for a geographical subsystem being assigned to each processor.

7. CONCLUSIONS

Developing scalable applications is necessary as power systems expand, interconnection gets denser, and newer equipment gets added. This paper discussed the high-performance computing library PETSc as a potential platform for rapid

development of existing and future power system applications. The development of PETSc has been funded by the Department of Energy for over 15 years. Because of its wide use among DOE application scientists, its continued long term development and support is highly likely.

8. 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.

9. REFERENCES

- [1] Balay, S., J. Brown, K. Buschelman, V. Eijkhout, W. Gropp, D. Kaushik, M. Knepley, L. McInnes, and B. F. Smith, and H. Zhang, *PETSc users manual*. ANL-95/11-3.1 (2010).
- [2] Balay, S., J. Brown, K. Buschelman, V. Eijkhout, W. Gropp, D. Kaushik, M. Knepley, L. McInnes, and B. F. Smith, and H. Zhang, *PETSc users manual*. ANL-95/11-3.2 (May 2011).
- [3] Balay, S., J. Brown, K. Buschelman, V. Eijkhout, W. Gropp, D. Kaushik, M. Knepley, L. McInnes, and B. F. Smith, and H. Zhang, *PETSc Web page*. (2011) <http://www.mcs.anl.gov/petsc>.
- [4] Knepley, M., "PETSc Tutorial". *Short Course on Scientific Computing* Chinese Academy of Sciences, Beijing, China. (2010). <http://www.mcs.anl.gov/petsc/petsc-as/documentation/tutorials/GUCASTutorial10.pdf>
- [5] Benson S., L. C. McInnes, J. Moré, T. Munson, and J. Sarich", "*TAO User Manual (Revision 1.10.1)*", ANL/MCS-TM-242 (2010).
- [6] Roman, J. E., E. Romero, and A. Tomas, *SLEPc users manual* DSIC-II/24/02 - Revision 3.1, D. Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, (2010).
- [7] Abhyankar, S. G. *An implicitly coupled electromechanical and electromagnetic transients simulator for power systems*. Ph.D. Dissertation, 2011.
- [8] Falcao., D., "High performance computing in power system applications", *Invited Lecture at the 2 International Meeting on Vector and Parallel Processing*, Porto, Portugal, (1996).
- [9] Grijalva S., "Research needs in multi-dimensional, multi-scale modeling and algorithms for next generation electricity grids". *DOE conference on computing challenges for the next-generation power grid* (2011).
- [10] IEEE Task Force of the Computer and Analytical Subcommittee of the Power Systems Engineering Committee. "Parallel processing in Power Systems Computation", *IEEE Transactions on Power Systems* 7.2 (May 1992).
- [11] Zhenyu H., Y. Chen, and J. Nieplocha. "Massive contingency analysis with high performance computing", *IEEE Power and Energy Society General Meeting* (July 2009).
- [12] Zhenyu H., and J. Nieplocha, "Transforming Power Grid Operations via High-Performance Computing", *IEEE Power and Energy Society General Meeting 2008* Pittsburgh, PA, USA, (July 20-24, 2008).

- [13] Smith, B., and H. Zhang, “Sparse Triangular Solves for ILU Revisited: Data Layout Crucial to Better Performance”, *International Journal of High Performance Computing Applications*, 2010. DOI 10.1177/10 94342010384857.
- [14] Chai., J. S., N. 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*, 6.4 (November 1991): 9-15.
- [15] Decker, I. C., D. M. Falcao, and E. Kaszkurewicz. “Conjugate gradient methods for power system dynamic simulation on parallel computers”, *IEEE Transactions on Power Systems*. 9.2 (May 1994): 629-636.
- [16] Alvarado, F. L., “Parallel solution of transient problems by trapezoidal integration”, *IEEE Transactions on Power Apparatus and Systems*. PAS-98 (May/June 1979): 1080-1090.
- [17] Ilic., M., M. L. Crow, and M. A. Pai., “Transient stability simulation by waveform relaxation methods”, *IEEE Transactions on Power Systems*. 2.4 (November 1987): 943 - 952.
- [18] Crow., M. L., and M. Ilic., “The parallel implementation of waveform relaxation methods for transient stability simulations”, *IEEE Transactions on Power Systems*. 5.3 (August 1990): 922 - 932.
- [19] Hou, L., and A. Bose., “Implementation of the waveform relaxation algorithm on a shared memory computer for the transient stability problem”, *IEEE Transactions on Power Systems*. 5.3 (August 1991).
- [20] Jalili-Marandi V., and V. Dinavahi., “SIMD-based large scale transient stability simulation on the graphics processing unit”, *IEEE Transactions on Power Systems*. 20.3 (August 2010): 1589 - 1599.
- [21] IEEE/CIGRE Joint Task Force on Stability Terms and Definitions., “Interfacing Techniques for transient stability and electromagnetic transients program”, *IEEE Transactions on Power Apparatus Systems*. 8.4 (October 2009): 2385-2395.
- [22] Sauer, P.W., and M.A.Pai., *Power System Dynamics and Stability*, New Jersey: Prentice Hall Inc., 1998.
- [23] Watson, N., and J. Arrillaga., *Power System Electromagnetic Transients Simulation*, London,UK: The Institution of Electrical Engineers, 2003.

The submitted manuscript has 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.