



LAWRENCE  
LIVERMORE  
NATIONAL  
LABORATORY

# A Parallel Multigrid Reduction in Time Method for Power Systems

M. Lecouvez, R. Falgout, C. Woodward, P. Top

November 10, 2015

Power & Energy Society General Meeting  
Boston, MA, United States  
July 17, 2016 through July 21, 2016

## **Disclaimer**

---

This document was prepared as an account of work sponsored by an agency of the United States government. Neither the United States government nor Lawrence Livermore National Security, LLC, nor any of their employees makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States government or Lawrence Livermore National Security, LLC. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States government or Lawrence Livermore National Security, LLC, and shall not be used for advertising or product endorsement purposes.

# A Parallel Multigrid Reduction in Time Method for Power Systems

Matthieu Lecouvez, Robert D. Falgout, Carol S. Woodward, Philip Top  
Lawrence Livermore National Laboratory  
Livermore, California 94550  
Email: {lecouvez1, falgout2, woodward6, top1}@llnl.gov

**Abstract**—This paper presents a fully multilevel approach to parallel in time solution of transient power system simulations. The method employs a multigrid reduction algorithm in time parallelized using the MPI distributed memory programming model. The method is demonstrated on a simple *Single Machine Infinite Bus* differential-algebraic equation model problem, for which speedup is obtained for as few as 8 processing cores on a problem with 10,000 time steps. Speedup of a factor of 13 is observed for a 100,000 step version of this simple problem. Based on these results, we expect significantly better speedup on larger problems where more work is available to each processor allowing greater amortization of the parallel communication.

## I. INTRODUCTION

Time-domain simulations of power system dynamics are currently used for a number of different purposes. Dynamic contingency analysis examines a few seconds after some event or sequence of events, though many such events might be tested. Small signal stability tests the response of a system to small stimuli checking for instabilities. Other applications look at the control actions and system response. Renewable energy ramps and cascading events require simulations over the course of minutes or even hours. For some of the applications faster than real-time simulations are necessary for timely response to conditions, and to test or explore the impact of real-time control actions. In other applications faster simulation time would allow for higher resolution models or additional scenarios. Significant speedups to dynamic power system simulation could have an important impact on the way the grid is controlled and could enable new applications and uses inside grid control facilities, particularly if a large enough model could be simulated faster than real-time.

High performance parallel computers can enable significant speedups in these simulations. One strategy for parallelization of time-domain simulations has been to parcel out the system components over the processing cores (spatial decomposition) [1], [2]. The main challenge with this strategy is the development of an effective distributed memory parallel linear solver. These solvers are an active area of research [3], [4], and no solver has yet proven to be fully effective.

Another strategy for parallelization of these systems is to decompose time into subintervals and distribute these over the processing cores. This parallel-in-time approach is also an active area of research in many fields where it has been shown to give substantial speedup. The first significant method of this kind for power systems was developed in [5], [6].

It used a relaxation/Newton method where groups of time steps were solved together as a coupled system, and multiple groups were solved in parallel. The relaxation was done on a sequence of finer and finer temporal grids in a multilevel nested iteration approach. More recently, researchers have looked at the *parareal* algorithm which is a two-level parallel-in-time method. In [7], the parareal algorithm was applied to a partitioned solution method for transient power systems. In that work, the parallel-in-time method was shown to be effective compared to the relaxation Newton method.

In this paper, we study a different approach to the parallel-in-time method called multigrid reduction in time [8]. Multiple levels of coarsening in time are used, and the method is applied to the full *differential algebraic equation* (DAE), not just the differential portion within a partitioned approach. We first describe the method in more detail and discuss its implementation based on the open-source code XBraid [9]. We then provide parallel performance results for a model problem and finish with conclusions and plans for future work.

## II. PARALLEL IN TIME METHOD

Research on parallel-in-time methods started with the work of Nievergelt in 1964 [10]. Since then, a variety of approaches have been developed, including the parareal method mentioned in the introduction [11]. However, relatively little work has been done in this area overall, especially considering that more than 50 years have passed since it was first explored. For a recent review of the literature, see [12]. The method considered here is called multigrid reduction in time (MGRIT) [8]. It is based on multigrid reduction techniques [13], [14] and is relatively non-intrusive on existing codes. When restricted to two grid levels, it is equivalent to parareal.

Let  $u(t)$  be the solution to some time-dependent problem (for example, a DAE) on time interval  $[0, T]$ . Let  $t_i = i\delta t, i = 0, 1, \dots, N$  be a temporal mesh on that interval with constant spacing  $\delta t = T/N$ , and let  $u_i$  be an approximation to  $u(t_i)$ . A one-step time discretization is then given by

$$u_0 = g_0, \quad u_i = \Phi_i(u_{i-1}) + g_i, \quad i = 1, 2, \dots, N. \quad (1)$$

A traditional time stepping method solves for  $u_1$  through  $u_N$  in sequence. To motivate the MGRIT approach, first consider the simple linear case where each function  $\Phi_i$  is a matrix.

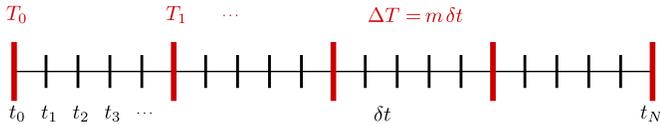


Fig. 1. Fine grid ( $t_i$ ) and coarse grid ( $T_j$ ) for coarsening factor  $m = 5$ . The coarse grid induces a decomposition of the fine grid into  $C$ -points (red) and  $F$ -points (black).

Then time stepping is equivalent to a forward solve of the block linear system  $\mathbf{A}\mathbf{u} = \mathbf{g}$  given by

$$\begin{pmatrix} I & & & & \\ -\Phi_1 & I & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & -\Phi_N & I \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_N \end{pmatrix} = \begin{pmatrix} g_0 \\ g_1 \\ \vdots \\ g_N \end{pmatrix}. \quad (2)$$

The idea is to replace the  $O(N)$  sequential time stepping method with an  $O(N)$  *multigrid* [15] iterative solver that is highly parallel. One well-known direct method for solving tridiagonal systems is *cyclic reduction*. The MGRIT multigrid method is a kind of approximate block cyclic reduction algorithm that utilizes a sequence of coarser temporal systems to accelerate the solution of the fine grid problem in (2). Although the above motivation assumes a linear system, the algorithm applies to the full nonlinear setting. It is also important to note that we solve the same discrete space-time system as in (1); that is, MGRIT converges to the solution produced by sequential time stepping on the finest grid.

The coarse grids in MGRIT are formed from the original fine grid by successively coarsening with factor  $m > 1$ . The coarsening of a grid induces a decomposition of that grid into two sets called  $C$ -points (points that align with the coarse grid) and  $F$ -points (everything else). Figure 1 provides an illustration in the case of the finest grid level. With this decomposition in hand, we can describe *relaxation* and *coarse-grid correction*, the two main components of multigrid.

Relaxation alternates between so-called F-relaxation and C-relaxation. F-relaxation updates the  $F$ -point values  $u_i$  on interval  $(T_j, T_{j+1})$  by propagating the  $C$ -point value  $u_{mj}$  across the interval using the time propagators  $\Phi_i$  in sequence. Although this is a sequential process, the F-intervals are independent from each other and can be computed in parallel. Similarly, C-relaxation updates the  $C$ -point value  $u_{mj}$  based on the  $F$ -point value  $u_{mj-1}$ , and these updates can also be computed in parallel. From this, so-called FCF-relaxation is just a composition of successive F-, C-, and F-relaxations. Simple injection is used to transfer values between grids (Step 2 and Step 5 below).

The two-grid MGRIT algorithm is then as follows:

- 1) Apply FCF-relaxation to  $A(\mathbf{u}) = \mathbf{g}$ .
- 2) Restrict the fine grid approximation and residual to the coarse grid:  $u_{\Delta,j} \leftarrow u_{mj}$ ,  $r_{\Delta,j} \leftarrow g_{mj} - A(\mathbf{u})_{mj}$ .
- 3) Solve  $A_{\Delta}(\mathbf{v}_{\Delta}) = A_{\Delta}(\mathbf{u}_{\Delta}) + \mathbf{r}_{\Delta}$ .
- 4) Compute coarse error approximation:  $\mathbf{e}_{\Delta} = \mathbf{v}_{\Delta} - \mathbf{u}_{\Delta}$ .

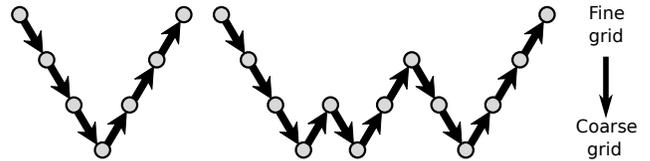


Fig. 2. Multigrid V-cycle (left) and F-cycle (right). Cycles represent the order in which different time grids are visited in the recursive MGRIT algorithm. Going down to a coarser grid corresponds to step 2 of the MGRIT algorithm, while going up corresponds to step 5.

- 5) Correct  $\mathbf{u}$  at  $C$ -points:  $u_{mj} = u_{mj} + e_{\Delta,j}$ .
- 6) Apply F-relaxation.

The multilevel algorithm uses the two-grid method in a recursive fashion to solve the system in Step 3. A variety of standard multigrid cycling strategies may be applied, including V-cycles and F-cycles (see Figure 2). For a fairly comprehensive reference on multigrid methods and techniques, see [15]. One important aspect of MGRIT is that the user only needs to define  $\Phi_i$ , which corresponds to the original time stepping method. Hence, most of the original code can be used as is, making the method relatively non-intrusive.

It is insightful to compare and contrast the MGRIT approach to some of the previous work done for power systems. In the interest of space, we mention only two methods here, one early [5], [6] and the other more recent [7].

The method introduced in [5] was well ahead of its time and was one of the first to demonstrate potential for speeding up power grid simulations with a parallel-in-time approach. Like MGRIT, the algorithm uses a sequence of coarse grids and coarse-grid problems. It does nested iteration in an effort to provide better and better initial guesses on each grid level, starting on the coarsest grid and moving up the hierarchy to the finest grid. A variety of relaxation/Newton methods were studied as the iterative solver on each grid level [6]. MGRIT differs primarily through the FCF-relaxation scheme and the different grid cycling strategies used. In general, nested iteration with relaxation will not produce an optimal  $O(N)$  algorithm and more complex cycling strategies such as F-cycles (Figure 2) are needed.

The method in [7] is based on the popular parareal algorithm [11]. Like MGRIT, parareal is also non-intrusive and easy to integrate with existing serial time-stepping code, and as mentioned above, it is equivalent to MGRIT when using only two time grids and just F-relaxation. The work in [7] demonstrates the potential for speedup in power grid simulations by running serial code and assuming that parallel communication costs are negligible. In general, scalability of multigrid algorithms in parallel requires more than two grid levels. A more extensive discussion of this along with detailed parallel performance analyses can be found in [8], [16].

### III. IMPLEMENTATION

In this section, we detail the implementation of the MGRIT algorithm for power grid problems. Power grid simulation

involves the solution of differential algebraic equations (DAE), whose most general formulation is

$$F(t, y, \dot{y}) = 0, \quad y(0) = y_0, \quad (3)$$

where the Jacobian  $\frac{\partial F}{\partial \dot{y}}$  may be singular. Implicit Runge-Kutta methods are used as time integrators. An  $s$ -stage Runge-Kutta method is given by its set of coefficients  $(A_{i,j})$ ,  $(b_i)$  and  $(c_j)$  for  $i, j = 1..s$ . Let  $y_n$  represent an approximation of  $y(t_n)$ . Then the computation of  $y_n$  from the previous step  $y_{n-1}$  is given by  $y_n = y_{n-1} + h \sum_{i=1}^s b_i K_i$  where  $K_i$  are solutions of the following nonlinear problem (see [17])

$$F \left( t_{n-1} + c_i h, y_{n-1} + h \sum_{j=0}^s a_{i,j} K_j, K_i \right) = 0, \quad (4)$$

for  $i = 1, \dots, s$ . Although not necessary, for this paper we made the choice to use the same time integrator for each grid level. This integrator is a diagonally-implicit, five-stage and fourth-order Runge-Kutta method, developed by Cash in [18]. The nonlinear system arising at each time step is solved by a Newton solver and requires the evaluation of the Jacobians  $\frac{\partial F}{\partial y}$  and  $\frac{\partial F}{\partial \dot{y}}$ . They are evaluated with finite differences and updated only once at the beginning of each time step. The relative tolerance required for the Newton solver is  $10^{-8}$  and all linear systems are solved by a direct solver (Lapack LU factorization). All these methods are implemented in C++.

The implementation of our parallel MGRIT solver was done via the open source software library XBraid [9], which was designed to be relatively non-intrusive on existing codes. Using it mainly requires writing a step function that produces the action of  $\Phi_i$  in (1), and that step function is usually just a wrapper around the original time stepping routine. In our case, the step function solves the nonlinear problem (4).

To save on memory, XBraid only stores solution values at  $C$ -points. It also employs techniques to overlap communication and computation in each of the major components of the MGRIT algorithm (e.g., relaxation). This is done by first posting a non-blocking receive from the processor to the left, then beginning computation with the right-most  $F$ -interval so that a non-blocking send can be posted as soon as possible to the right processor. While this communication is completing, computation is done on the interior of the processor's interval.

#### IV. RESULTS

We present here our first results of the parallel-in-time algorithm for power system simulation. We focus on a fairly basic model problem. The problem is an order 2 *Single Machine Infinite Bus* (SMIB) model and consists of six state variables. More details about the model can be found in [19]. The model equations are

$$\begin{cases} \dot{\delta} = \Omega_b (\omega - 1) \\ \dot{\omega} = \frac{1}{2H} (P_m - P_e - D (\omega - 1)) \\ 0 = v_q + R_s i_q - e'_q + (x'_d - x_l) i_d \\ 0 = v_d + R_s i_d - (x'_d - x_l) i_q \\ 0 = P_l + \frac{V}{x_d} \sin \theta - P_G \\ 0 = Q_l - \frac{V^2}{x_d} + \frac{V}{x_d} \cos \theta - Q_G \end{cases}, \quad (5)$$

TABLE I

VALUE OF THE PARAMETERS USED FOR THE NUMERICAL SIMULATIONS

$\Omega_b$	$H$	$D$	$R_s$	$P_m$
$120\pi$	5.0	4.0	0.05	1.2
$P_l$	$Q_l$	$x_d$	$x'_d$	$x_l$
0.2	-0.3	1.05	0.35	0.0

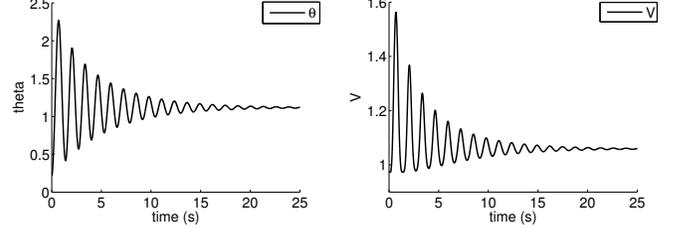


Fig. 3. Two components of the solution over the time interval  $[0, 25]$ .

where the intermediate variables are defined by

$$\begin{cases} P_e = (v_q + R_s i_q) i_q + (v_d + R_s i_d) i_d \\ P_G = v_q i_q + v_d i_d \\ Q_G = v_q i_d - v_d i_q \\ v_q = V \cos(\delta - \theta) \\ v_d = V \sin(\delta - \theta) \end{cases}. \quad (6)$$

Two of the unknowns are differential variables ( $\delta$  and  $\omega$ ) and correspond to the angle and rotational speed of the generator. The last four are algebraic variables ( $i_q$ ,  $i_d$ ,  $V$  and  $\theta$ ). Both  $i_q$  and  $i_d$  refer the currents in the generator, and  $V$  and  $\theta$  represent the voltage and phase angle of the bus. The system as a whole represents how a simple generator responds to local variations while connected to a stiff network represented by the infinite bus. This model includes several parameters whose values are summarized in Table I. Most of the parameters relate to the generator model, ( $x_d$ ,  $x'_d$ , and  $x_l$ ) are the generator impedances,  $R_s$  is the winding resistance,  $D$  is a damping coefficient,  $P_m$  is the mechanical power input to the generator, and  $H$  is an inertial constant. The base frequency is  $\Omega_b$  and the real and reactive load is  $P_l$  and  $Q_l$ . The initial values of the unknowns at time  $t = 0$  are determined as follows:

- $\omega(0) = 1$  and  $\dot{\omega}(0) = 0$
- $P_G(0) = 2P_l$  and  $Q_G(0) = Q_l$
- All equations in (5) are satisfied, except the third.
- The third equation in (5) determines the value of  $e'_q$ .

Note that with these values of parameters, the model remains valid, i.e. the generator stays within a stable regime. No partitioning technique is used here, and the MGRIT algorithm is applied to the full DAE system. Some components of the solution are represented in Figure 3. The solution is characteristic of a sudden load reduction on the system at time 0. The generator output power oscillates from the change in loading and the oscillations are slowly damped settling on a new equilibrium with a higher voltage and a further leading angle, as one expects from this situation.

The scalability of the algorithm is studied for two step sizes and intervals of integration. In the first case, the system is integrated over the interval  $[0, 10s]$ , and 10,000 points in time are used on the finest grid. These values lead to a time-step size of 1 ms. This simulation is clearly not computationally expensive since it runs sequentially in about one second. However, the simulation duration and time step size are similar to those used when running contingency analysis scenarios. We see that some speedup can still be achieved on this problem indicating this method will allow full use of a large-scale machine when one has more processing cores than scenarios. To highlight the potential of this method, the second test case we study corresponds to the integration of the system over the interval  $[0, 50]$  with 100,000 points in time.

The scalability of the MGRIT method depends on several parameters, including the type of cycle used, the number of levels, and the coarsening factor. Since a sequential run is executed on the coarsest grid in XBraid, having many levels improves scalability. On the other hand, increasing the number of levels increases the cost of an iteration since more levels implies more function evaluations. One way to overcome this cost issue is to use a larger coarsening factor, thus reducing the cost of the coarse grid. The number of levels is also limited by the stability of the time integrator. It must remain stable on the coarser grid, although alternate integrators with greater stability can be applied on the coarse grids. For our experiments, we used F-cycles (Figure 2). Recall that the MGRIT algorithm converges to the initial sequential time stepping algorithm, so all previously mentioned parameters do not influence the accuracy of the method. The only parameter governing the accuracy of the method is the absolute tolerance used to stop the iterations. Here we took  $10^{-8}$  as the stopping criteria for all simulations.

All computational experiments were run on the machine ‘‘Cab’’ at Lawrence Livermore National Laboratory. This machine is an Intel Xeon-based system with over 1,200 compute nodes, each having two 8-core CPUs in a shared memory configuration. Parallel communications are made through MPI (Message Passing Interface).

#### A. 10s simulation, 10,000 time points

Table II shows, for various coarsening factors, the maximum number of levels used in the MGRIT algorithm and the minimum number of processors required to overcome the additional cost of the multigrid method relative to time stepping (crossover point). One can see that the required number of processors to reach the sequential time decreases with the coarsening factor. This behavior was expected since a large coarsening factor reduces the additional cost of the multigrid method. Figure 4 represents the time to solution as a function of the number of processors. The horizontal dashed line is the reference (sequential) time to solution. The crossover point corresponds to the intersection with this reference time, and from this point, speedup can be achieved. The maximum speedup we obtained for this very small case is about 5. This speedup is achieved sooner (at a smaller processor count) with

TABLE II  
COARSENING FACTOR AND NUMBER OF PROCESSORS TO REACH CROSSOVER POINT WITH SEQUENTIAL RUN

Coarsening factor	2	3	4	5
Max number of levels	7	4	3	3
Crossover point	18	8	6	5

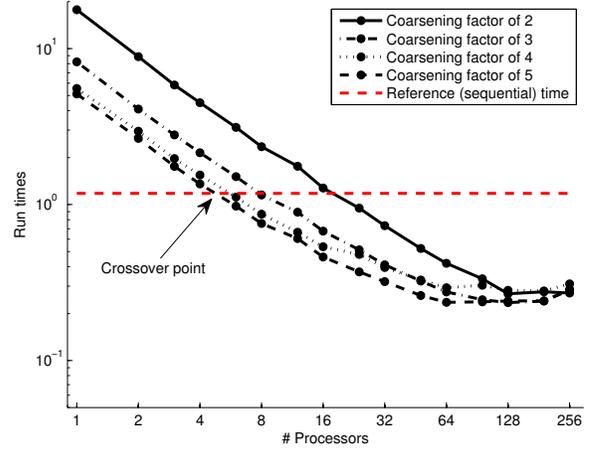


Fig. 4. Evolution of the time to solution with respect to the number of processors. Best achievable speedup is about 5 for this 10,000 point example.

a large coarsening factor. In our example, 64 processors with a coarsening factor of 5 produced the best speedup, but a speedup was also observed even with 8 processors. Beyond 128 processors, the run time stagnates and no more speedup can be obtained. At this point, communication between processors becomes preponderant (there are only  $\sim 80$  time points per processor on the finest grid).

Even though the problem is quite small, some speedup can be obtained with this parallel-in-time technique. Moreover, up to a certain limit, XBraid shows good scalability. Note that for this problem, the convergence of the multigrid process is extremely fast, and only two iterations are needed to reach the desired precision ( $10^{-8}$ ).

#### B. 50s simulation, 100,000 time points

In this second test case, we increase the number of points in time, as well as the duration of integration. On the finest grid, the time step size is now 0.5 ms. The sequential time to solution for this problem is about 10 seconds. From Table III, one can see that the crossover point remains almost unchanged from the previous case. Moreover, the scalability of the algorithm is improved. Figure 5 shows the time to solution for coarsening factors of 2 and 4 with different numbers of processors. The achieved speedup is about 13 in this case. Again, at some point communication costs take precedence over computation costs, leading to stagnation in the total run time. For this case, it happens around 512 processors.

TABLE III  
COARSENING FACTOR AND NUMBER OF PROCESSORS TO REACH  
CROSSOVER POINT WITH SEQUENTIAL RUN

Coarsening factor	2	3	4	5
Max number of levels	8	5	4	3
Crossover point	20	9	7	6

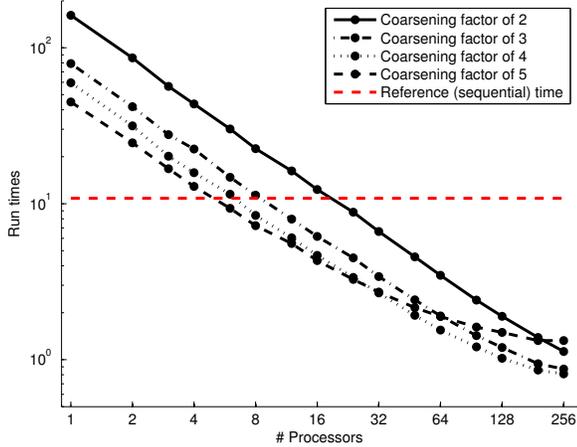


Fig. 5. Evolution of the time to solution with respect to the number of processors. Best achievable speedup here is about 13. Problem size is 100, 000 points in time.

## V. CONCLUSIONS

In this paper, we presented a new approach for parallelizing power system simulation. This approach uses a multigrid reduction technique to achieve parallelism in the time dimension. This allows a significant speedup to be achieved compared to sequential time stepping. In addition, it enables the use of many more processors than when parallelizing only in space. This is the direction future computer architectures are taking. We used the XBraid implementation of the MGRIT algorithm as a non-intrusive library. First results on a very basic test case show that speedups can be obtained even on small problems as are common in contingency analysis. Although both the serial and parallel methods implemented here were our own, the speedups are indicative of what can be achieved by using XBraid and the MGRIT algorithm together with an existing production power grid code. The strong scaling of the method would likely be slightly worse, because a more optimized code would result in relatively larger communication costs. On the other hand, production problems also involve more work per processor, which would have the opposite effect. Next steps include implementing and benchmarking more complex scenarios to further assess potential benefits and drawbacks of this method for power system simulations. Although the results here involved a relatively small problem, experience with much larger PDE problems on up to 65K cores [8], [16] supports the case that good parallel performance on larger power grid systems is achievable. Convergence of MGRIT on these systems, however, is a more open research question.

## ACKNOWLEDGMENT

This work was supported by the United States Department of Energy Office of Electrical Delivery and Energy Reliability Advanced Grid Modeling Program. This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC. LLNL-CONF-679148.

## REFERENCES

- [1] W. Hatcher, F. Brasch Jr, and J. Van Ness, "A feasibility study for the solution of transient stability problems by multiprocessor structures," *IEEE Trans. Power App. Syst.*, vol. 96, no. 6, pp. 1789–1797, 1977.
- [2] J. Shu, W. Xue, and W. Zheng, "A parallel transient stability simulation for power systems," *IEEE Trans. Power Syst.*, vol. 20, no. 4, pp. 1709–1717, 2005.
- [3] S. Abhyankar, B. Smith, and E. Constantinescu, "Evaluation of overlapping restricted additive Schwarz preconditioning for parallel solution of very large power flow problems," in *Proceedings of the 3rd International Workshop on High Performance Computing, Networking and Analytics for the Power Grid*. ACM, 2013, p. 5.
- [4] S. K. Khaitan and J. D. McCalley, "A class of new preconditioners for linear solvers used in power system time-domain simulation," *IEEE Trans. Power Syst.*, vol. 25, no. 4, pp. 1835–1844, 2010.
- [5] M. La Scala, A. Bose, D. J. Tylavsky, and J. S. Chai, "A highly parallel method for transient stability analysis," *IEEE Trans. Power Syst.*, vol. 5, no. 4, pp. 1439–1446, 1990.
- [6] M. La Scala and A. Bose, "Relaxation/Newton methods for concurrent time step solution of differential-algebraic equations in power system dynamic simulations," *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on*, vol. 40, no. 5, pp. 317–330, 1993.
- [7] G. Gurrula, A. Dimitrovski, S. Pannala, S. Simunovic, and M. Starke, "Parareal in time for fast power system dynamic simulations," *IEEE Trans. Power Syst.*, to appear, DOI: 10.1109/TPWRS.2015.2434833.
- [8] R. D. Falgout, S. Friedhoff, T. V. Kolev, S. P. MacLachlan, and J. B. Schroder, "Parallel time integration with multigrid," *SIAM J. Sci. Comput.*, vol. 36, no. 6, pp. C635–C661, 2014, LLNL-JRNL-645325.
- [9] "XBraid: Parallel multigrid in time," <http://llnl.gov/casc/xbraid>.
- [10] J. Nievergelt, "Parallel methods for integrating ordinary differential equations," *Comm. ACM*, vol. 7, pp. 731–733, 1964.
- [11] J. L. Lions, Y. Maday, and G. Turinici, "Résolution d'EDP par un schéma en temps pararéel," *C.R.Acad. Sci. Paris Sér. I Math*, vol. 332, pp. 661–668, 2001.
- [12] M. J. Gander, "50 years of time parallel time integration," in *Multiple Shooting and Time Domain Decomposition Methods*, T. Carraro, M. Geiger, S. Körkel, and R. Rannacher, Eds. Springer Verlag, 2015, pp. 69–114.
- [13] M. Ries and U. Trottenberg, "MGR-ein blitzschneller elliptischer löser," Universität Bonn, Tech. Rep. Preprint 277 SFB 72, 1979.
- [14] M. Ries, U. Trottenberg, and G. Winter, "A note on MGR methods," *Linear Algebra Appl.*, vol. 49, pp. 1–26, 1983.
- [15] U. Trottenberg, C. Oosterlee, and A. Schüller, *Multigrid*. San Diego: Academic Press, 2001.
- [16] R. D. Falgout, S. Friedhoff, T. V. Kolev, S. P. MacLachlan, J. B. Schroder, and S. Vandewalle, "Multigrid methods with space-time concurrency," *SIAM J. Sci. Comput.*, submitted, LLNL-JRNL-678572.
- [17] U. Ascher and L. Petzold, *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. SIAM, 1998.
- [18] J. R. Cash, "Diagonally implicit Runge-Kutta formulae with error estimates," *IMA Journal of Applied Mathematics*, vol. 24, no. 3, pp. 293–301, 1979.
- [19] J. Machowski, J. W. Bialek, and J. R. Bumby, *Power Systems Dynamics Stability and Control*, 2nd ed. "The Atrium, Southern Gate, Chichester, West Sussex, PO19 8SQ": John Wiley and Sons, Ltd, 2008.