Parallel-in-Time Solution of Power Systems with Scheduled Events

Jacob B. Schroder, Robert D. Falgout,
Carol S. Woodward, and Philip Top
Lawrence Livermore National Laboratory
Livermore, California 94550
Email: {schroder2, falgout2, woodward6, top1}@llnl.gov

Matthieu Lecouvez
Centre d'études scientifiques et techniques d'Aquitaine, CEA/Cesta
15 Avenue des Sablières, 33114 Le Barp, France
Email: matthieu.lecouvez@cea.fr

***Abstract*—This work presents a time-parallel approach for solving power grid problems with scheduled events (e.g., a square pulse applied to a bus). The multigrid reduction in time (MGRIT) method, implemented in the XBraid code, is used to nonintrusively add time parallelism to GridDyn, an existing open-source power grid simulation tool. The approach extends standard practices for sequential time stepping and scheduled events to the time-parallel setting and then explores temporal refinement. The paper concludes with parallel experiments validating the approach and showing speedups of around 50x for the WECC 179 bus system. Scalability with respect to scheduled events is examined by applying a one second long square pulse to bus 143 every 2 seconds.**

I. Introduction

Time-domain simulations of power system dynamics are currently used in numerous areas, including dynamic contingency analysis, small signal stability analysis, control and system response analysis, renewable planning, and cascading event analysis. The latter two uses require simulations over the course of minutes or even hours. For some applications, faster than real-time simulations are necessary, but others require faster simulation time to allow for higher resolution models or additional scenarios. For both the cases of long simulation times and desired faster computations, significant code speedups over current technologies are needed.

High performance parallel computers (HPC) can enable these speedups. One strategy for using HPC has been to parallelize the computation over system components [1], [2], but this strategy poses a large challenge in development of effective distributed memory parallel linear system solvers. 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 subintervals over the processors. This parallel-in-time approach is also an active area of research in many fields, and it has been shown to give substantial speedup. This method was first used for power systems in the relaxation/Newton method developed by [5], [6] where multiple groups of coupled time step solutions were solved in parallel, and relaxation was done on a sequence of finer and finer temporal grids in a multilevel nested iteration approach. More recently, researchers have explored the *parareal* algorithm which is a two-level parallel-in-time method. In [7], the parareal algorithm was shown to be effective compared to the relaxation Newton method. In [8], a simpler reduced model was used for the coarse propagator in an effort to decrease computational cost and speed up simulation time within the parareal framework.

In this paper, we study a different parallel-in-time approach called multigrid reduction in time [9]. Here, multiple levels of coarsening in time are used, and the method is applied to the full *differential algebraic equation* (DAE) system, not just the differential portion within a partitioned approach. In the previous work [10], we applied our approach to a single machine infinite bus system on uniform time grids and demonstrated speedups of up to 13x over sequential time stepping. In [11], we extended the method from single-step multi-stage Runge-Kutta schemes to multi-step backward differencing (BDF) schemes on variably-stepped temporal grids applied to an IEEE 39 bus problem and a reduced WECC 179 bus problem. Moderate speedups were observed, depending on the order of the BDF discretization and the size of the temporal grid. In this paper, the main focus is on handling scheduled discontinuities, such as occur during load or generator output changes. We first describe the method and discuss its implementation based on the open-source code XBraid [12]. We then provide parallel performance results for the WECC 179 bus model problem, and finish with conclusions and plans for future work.

II. Parallel in Time Method

The parallel-in-time method used here is multigrid reduction in time (MGRIT) [9]. This method is based on multigrid reduction [13], and, when restricted to two levels, it is equivalent to parareal [14], perhaps the most popular parallel-in-time method. For a recent review of the literature, see [15].

We now give a brief overview of MGRIT and refer the reader to [9] for more details. Consider the one-step time discretization

$$u_i = \Phi_i(u_{i-1}) + g_i, \text{ for } i = 1, 2, ..., N \text{ and } u_0 = g_0, \qquad (1)$$

of some time-dependent process on the interval $[0, T]$. Partition the time domain as in Figure 1, where $t_i = i\delta t$ is the fine time grid, and $T_i = mi\delta t$ is the coarse time grid for coarsening factor m. Let u_i be an approximation to $u(t_i)$.

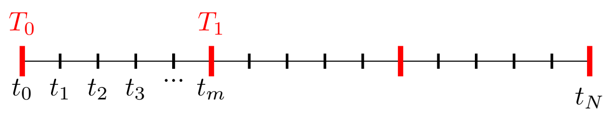

Fig. 1. Example fine and coarse level for a coarsening factor of $m = 5$. F-points (black) are present on only the fine level, whereas C-points (red) are on both the fine and coarse level.

Then, considering the linear case for simplicity, sequential time stepping is equivalent to a forward solve of the system $A\mathbf{u} = \mathbf{g}$,

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

where the solution $\mathbf{u}$ includes the state at all time points. MGRIT replaces this $O(N)$ but sequential solve of (2) with an iterative $O(N)$ solution using parallel multigrid reduction that simultaneously solves for all time solutions. As such, MGRIT converges to the same solution that serial time stepping produces. Additionally, the user need only define a routine that applies Φ, the map that advances the solution from one step to the next, making the method nonintrusive and allowing for reuse of serial time-stepping codes.

The new concurrency offered by MGRIT is made possible by coarse time grids which accelerate convergence to the solution on the original fine grid. Each coarse time grid is formed with an approximate block cyclic reduction strategy that eliminates block rows and columns in the system (2). If all the block rows corresponding to the black F-points in Figure 1 are eliminated, then a Schur-complement system, smaller by a factor of $1/m$, would result. This coarse system is approximated by a cheaper system that has the same form as (2) and uses a coarse time-stepping operator based on the larger coarse time-step size. Often, this new operator is taken to be just a rediscretization in time.

Complementing the coarse-grid corrections in MGRIT is relaxation which alternates between block Jacobi applied to the rows corresponding to F-points in (2) and then block Jacobi applied to the rows corresponding to C-points. FCF-relaxation corresponds to successive relaxation sweeps over F-points, then C-points, and then F-points again. Each relaxation sweep is a highly parallel process. Interpolation between time grids is done by injection.

This process is implemented as a full approximation storage (FAS) multigrid cycle [16], capable of solving nonlinear problems [17] and is described in [10]. The MGRIT algorithm uses standard multigrid cycling such as the V-cycles and F-cycles in Figure 2. F-cycles are more robust but scale less well in parallel because of the additional visits to coarse-levels.

Figure 3 depicts a cycle adapted to allow for variable time-step sizes through a user controlled temporal refinement strategy. Essentially, refinement in time occurs where the user's Φ routine reports a large temporal error. In the figure a V-cycle

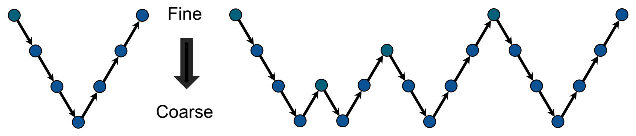

Fig. 2. Cycling strategies: V-cycle (left) and F-cycle (right).

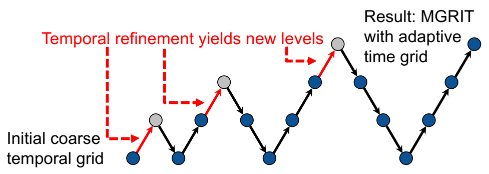
Temporal refinement yields new levels
Result: MGRIT with adaptive time grid
Initial coarse temporal grid

Fig. 3. Variable time-step strategy using F-cycles. The approach here is to begin on a coarse time grid, allow the user to specify which time intervals to refine, and then form a new level based on the newly refined time grid.

is carried out after each temporal refinement, although an F-cycle can also be carried out instead. This strategy is similar to nested iteration.

III. Event Handling

Our strategy for handling events that present as scheduled discontinuities in the DAE system extends standard practices for serial time stepping, i.e.,

1) Place a time point at each event location.
2) Allow for adaptive refinement in time on both sides of the event, using temporal error estimators.
3) Associate state information with each time point and retain this information on coarse grids. When taking a time step, state information associated with the starting time point is used. Hence, when taking a time step to the exact time value of an event, state information from before the event is used, and when taking a time step from the exact time value of an event, state information from after the event is used.

Interestingly, this strategy allows for a time point at an event to be coarsened away on coarser time grids. In addition, although state information may be discontinuous across event time values on the finest grid, discontinuities may not remain on coarser grids. As we will see, these changes in discontinuities on coarse grids do not degrade convergence.

The strategy used for extending MGRIT from one-step to multistep methods, such as BDF2, can be found in [11]. For a k-step method, k time points are lumped together, so that each group of time points depends only on the previous group, thus creating the appearance of a one-step scheme to MGRIT. However, this lumping can lead to instability in BDF methods through disparate time step sizes on coarse grids, and the solution used here is to coarsen in order to first-order BDF1, before coarsening in time.

IV. 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

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

where the Jacobian $\frac{\partial F}{\partial \dot{y}}$ may be singular. One of the time integration methods we consider is implicit Runge-Kutta. An s-stage Runge-Kutta method is defined by its set of coefficients $(A_{i,j})$, (b_i) and (c_j) for $i, j = 1, ..., s$. If y_n represents 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 the K_i satisfy the following nonlinear equations [18],

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

for $i = 1, ..., s$. Although not necessary, we use the same time integrator for each grid level.

In this work, we apply either a BDF2 implicit integrator or a diagonally-implicit, five-stage and fourth-order Runge-Kutta method, developed by Cash in [19]. For both methods, the nonlinear systems arising at each time step are solved by a Newton solver and require the evaluation of the Jacobians $\frac{\partial F}{\partial y}$ and $\frac{\partial F}{\partial \dot{y}}$. These Jacobians are directly evaluated and updated only once at the beginning of each time step. The linear systems are solved by a direct solver (Lapack LU factorization) and all methods are implemented in C++.

We used the nonintrusive open source software library XBraid [12] to implement our parallel MGRIT solver. Using XBraid mainly requires writing a step function that produces the action of Φ_i in (1), and that step function is usually just a wrapper around a preexisting time stepping routine. In our case, the step function solves the nonlinear problem (4) or a corresponding nonlinear problem for BDF2.

To provide memory savings, XBraid only requires solution values to be stored at C-points. It also employs techniques to overlap communication and computation in the various components of the MGRIT algorithm (e.g., relaxation). This overlap is achieved by first posting a non-blocking receive from the processor to the left (earlier in time), 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.

V. Results

This section describes the results of the proposed scheduled event handling approach for MGRIT. The model problem used is based on the WECC 179 bus system [20] with 793 unknowns, including 358 algebraic variables for the voltage and angles associated with the buses, plus 87 algebraic and 348 differential variables associated with the generators, with 4172 non-zero entries in the Jacobian. The generators are modeled using a 6th order machine model, with standard TGOV1 models for the governors and EXDC2 for the exciters. For this test the limits on the governors and exciters were disabled.

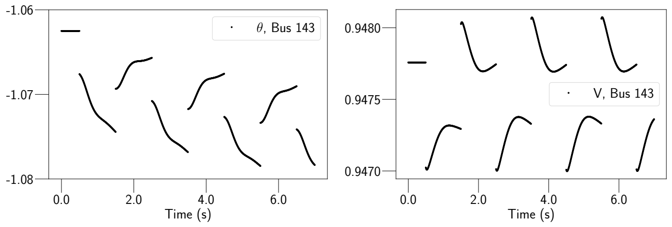

Fig. 4. Two solution components for bus 143 over $[0s, 7s]$.

The system was modified by adding a periodic square wave load to Bus 143. The period of the load was set to 2 seconds with the equivalent of a 50 MW load turning on and off at regular intervals. The simulation resulted in a series of sudden shifts in the loading and discontinuities in the algebraic variables of the solution. The computed voltages and angles at Bus 143 for the first few seconds are shown in Figure 4.

This simulation is intended to test the ability of the the parallel-in-time solver to handle scheduled discontinuities in the simulation, and the particular bus was chosen as a typical bus, rather than for any particular significance.

We now examine the scalability of XBraid for solving this problem by running strong scaling studies on Quartz, an Intel Xeon-based cluster at Lawrence Livermore, with 2 sockets and 36 cores per node.

Of particular interest is the scalability of XBraid as the number of events (i.e., discontinuities) increases. Scalability is explored by examining time-domain sizes of $t_f = 7s$, $57s$ and $460s$, with corresponding numbers of time steps of $N_t = 1,440$, $11,520$, and $92,160$, respectively. Recall that each second of simulated time encounters one event, thus we choose a longer final time value (here $460s$) to test a large number of events. Additionally, this large final time allows for XBraid to demonstrate its ability to offer large speedups.

The number of time steps was chosen to match the core counts on the machine, e.g., 256 nodes on Quartz corresponds to 9,216 cores, which in turn implies that $N_t = 92,160$ is a good choice for an XBraid coarsening factor of 10. In general, for an XBraid coarsening factor of k, placing k time points on each processor in time is an efficient strategy [9]. The final times are chosen so that the time step size $\delta t \approx 5$ms. The Newton solver used to take each time step uses a relative and absolute tolerance of 10^{-8}.

The XBraid parameters are as follows. We examine the use of F-cycles and V-cycles with FCF-relaxation. The stopping criteria is a residual tolerance of 10^{-8}. Thus, the relative difference between the XBraid solution and the sequential time stepping solution is less than 10^{-8}. All experiments coarsened to a time grid consisting of 4 points, thus making the method truly multilevel. Following the guidance of [10], we use a temporal coarsening factor of 10 for the implicit Runge-Kutta 4 method, and 5 for BDF2.

Three different time-stepping approaches are examined, the implicit RK4 scheme with a fixed time grid, BDF2 with a fixed time grid, and the implicit RK4 scheme with variable step sizes, i.e., temporal adaptivity. The variable time-stepping experiments use the adaptive cycling strategies discussed in Section II and Figure 3. We examine adaptivity with both V- and F-cycles. For the adaptivity, time intervals are refined based on an absolute and relative tolerance for the error estimator of 10^{-8}. The chosen implicit RK4 scheme provides a built-in estimator. The maximum allowed temporal refinement factor is 4. These parameters hold both for the sequential baseline experiments and XBraid. XBraid is allowed to refine the time grid until a grid with over 100,000 time points is reached, and sequential time stepping has its refinements capped with a minimum time-step size of 10^{-5}. These values were chosen so that the minimum δt in each case is comparable. Similar to [11], XBraid refines more than sequential time stepping, yielding 114,868 time steps, while sequential time stepping yields 104,513 time steps.

Table I depicts, for each fixed time grid problem, the maximum speedup achieved, XBraid iterations, and the first core count at which XBraid offers a speedup (cross-over). A '*' indicates that the experiment showed no cross-over, i.e., XBraid was always slower than sequential time stepping. We see that both the RK4 and BDF2 schemes on a fixed time grid show good scalability with respect to increasing numbers of discontinuities, i.e., the number of XBraid iterations increases only slightly.

TABLE I
XBRAID MAXIMUM SPEEDUP, CROSSOVER AND ITERATION COUNTS FOR THE FIXED TIME GRID STRONG SCALING STUDIES

	$t_f =$	7s	57s	460s
BDF2, V-cycles	Speedup	0.51	2.1	11.7
	Cross-over (core count)	*	4608	288
	XBraid iterations	21	28	30
BDF2, F-cycles	Speedup	0.50	1.95	9.1
	Cross-over (core count)	*	4608	288
	XBraid iterations	11	13	14
RK4, V-cycles	Speedup	2.5	10.5	53.3
	Cross-over (core count)	36	36	36
	XBraid iterations	7	9	10
RK4, F-cycles	Speedup	2.2	7.4	34.4
	Cross-over (core count)	36	36	36
	XBraid iterations	6	7	7

Figure 5 depicts the strong scaling study for the implicit RK4 method. The time to solution is shown as a function of the number of processors, with the horizontal dashed line representing the baseline time to solution for sequential time stepping. The exact cross-over point is the place where the baseline time to solution and the XBraid time to solution intersect. XBraid scales well, especially for V-cycles, with some degradation in scaling at the largest core counts. This degradation is due to communication costs beginning to dominate. We also see that V-cycles scale better than F-cycles at large core counts, eventually achieving faster runtimes, despite taking overall more iterations. This difference is because V-cycles have a lower per-iteration cost and have a superior communication complexity. We note that the speedup for the largest problem is 53.3x.

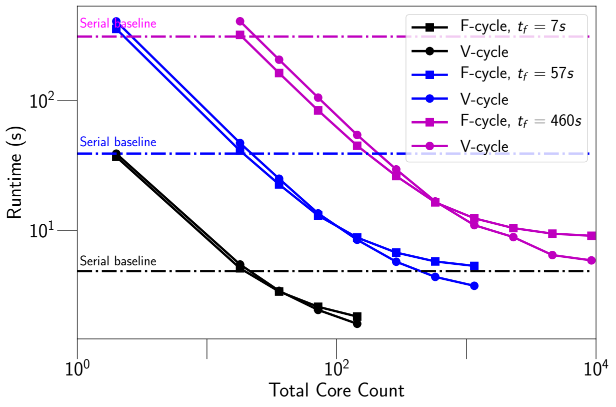

Fig. 5. Strong scaling study for three different problem sizes on a fixed time grid with $\delta t = 4.9$ms, using the implicit RK4 method.

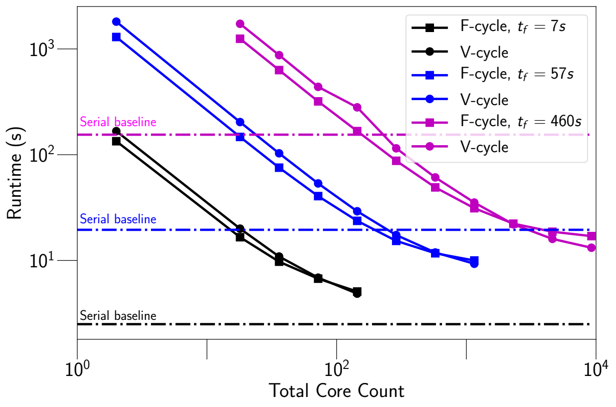

Fig. 6. Strong scaling study for three different problem sizes on a fixed time grid with $\delta t = 4.9$ms, using BDF2.

Figure 6 depicts the same strong scaling study, but for BDF2. We see similar strong scaling behavior with the maximum speedup being 11.7x. This speedup is less than that observed for the implicit RK4 scheme for two reasons. One, XBraid does not converge as fast for BDF schemes, requiring more overall iterations (see Table I). Two, the sequential BDF scheme is faster than the sequential implicit RK4 scheme, because each time step requires only one nonlinear solve.

Table II is analogous to Table I, but depicts results for the variable step implicit RK4 scheme with $t_f = 460s$. We see qualitatively the same robust convergence and speedup as in Table I, which confirms that the addition of variable time-stepping did not affect XBraid convergence.

TABLE II
XBRAID MAXIMUM SPEEDUP, CROSSOVER AND ITERATION COUNTS FOR THE VARIABLE STEP STRONG SCALING STUDIES

	Cycle	F	V
Var RK4, $t_f = 460s$	Speedup	27.3	46.8
	Cross-over (core count)	36	36
	XBraid iterations	7	9

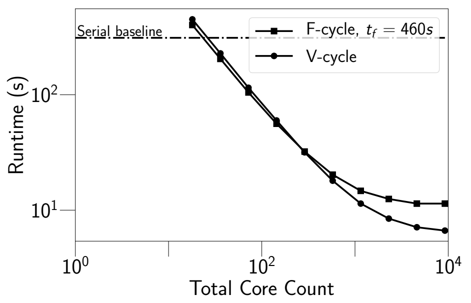

Fig. 7. Strong scaling for variable step sizes and the RK4 method.

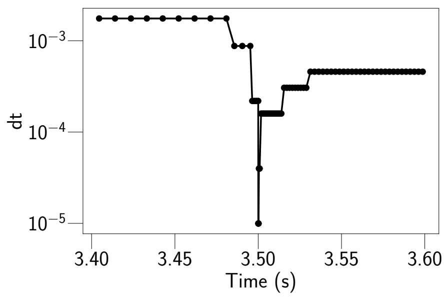

Fig. 8. Adaptive time grid found by XBraid, zoomed in around an event.

Figure 7 depicts the strong scaling study for the variable step implicit RK4 experiment with $t_f = 460s$. The maximum observed speedup is 46.8x. Figure 8 depicts the time-step sizes around the event at $t = 3.5$, showing the adaptive refinement.

VI. Conclusions

This work presents and validates an approach for handling scheduled events (i.e., discontinuities) for parallel-in-time simulations of power grid systems, using the XBraid framework. Overall, the approach extends standard practices for sequential time-stepping to the time-parallel setting for problems with discontinuous loads. Time points are placed at the location of the discontinuity and then adaptive temporal refinement is applied around these points for good discretization accuracy. When time-stepping to the discontinuity, state information before the discontinuity is used, and then when taking a step away from the discontinuity, the new state information, discontinuous from the previous time-point, is used.

The other main achievement is a demonstration of this approach on the WECC-179 system in parallel. The observed speedups are in the range of 11x for BDF2 and 53x for implicit RK4 (47x for adaptive time-stepping). We note that, to our knowledge, no other previous results for parallel-in-time methods and scheduled events with power grid systems exist. Future work will target the handling of unscheduled events.

Acknowledgment

This work was supported by the United States Department of Energy Office of Electrical Delivery and Energy Reliability Advanced Grid Modeling Program. This article has been authored by Lawrence Livermore National Security, LLC under Contract No. DE-AC52-07NA27344 with the U.S. Department of Energy. Accordingly, the United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this article or allow others to do so, for United States Government purposes. LLNL-CONF-740658.

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 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. Gurrala, A. Dimitrovski, S. Pannala, S. Simunovic, and M. Starke, “Parareal in time for fast power system dynamic simulations,” *”IEEE Trans. Power Syst.”*, vol. 31, no. 3, pp. 1820–1830, 2016.

[8] N. Duan, A. Dimitrovski, S. Simunovic, and K. Sun, “Applying reduced generator models in the coarse solver of parareal in time parallel power system simulation,” in *PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe), 2016 IEEE*. IEEE, 2016, pp. 1–5.

[9] 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.

[10] M. Lecouvez, R. D. Falgout, C. S. Woodward, and P. Top, “A parallel multigrid reduction in time method for power systems,” *Power and Energy Society General Meeting (PESGM)*, pp. 1–5, 2016.

[11] R. D. Falgout, M. Lecouvez, and C. S. Woodward, “A parallel-in-time algorithm for variable step multistep methods,” (submitted) 2017, LLNL-JRNL-739759.

[12] “XBraid: Parallel multigrid in time,” http://llnl.gov/casc/xbraid.

[13] M. Ries and U. Trottenberg, “MGR-ein blitzschneller elliptischer löser,” Universität Bonn, Tech. Rep. Preprint 277 SFB 72, 1979.

[14] 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.

[15] 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.

[16] A. Brandt, “Multi–level adaptive solutions to boundary–value problems,” *Math. Comp.*, vol. 31, no. 138, pp. 333–390, 1977.

[17] R. D. Falgout, A. Katz, T. V. Kolev, J. B. Schroder, A. Wissink, and U. M. Yang, “Parallel time integration with multigrid reduction for a compressible fluid dynamics application,” Lawrence Livermore National Laboratory, Tech. Rep. LLNL-JRNL-663416, 2015.

[18] U. Ascher and L. Petzold, *Computer Methods for Ordinary Differential Equations amd Differential-Algebraic Equations*. SIAM, 1998.

[19] J. R. Cash, “Diagonally implicit Runge-Kutta formulae with error estimates,” *IMA Journal of Applied Mathematics*, vol. 24, no. 3, pp. 293–301, 1979.

[20] “DC muti-infeed study,” Electric Power Research Institute, Tech. Rep. EPRI TR-104586s, Projects 2675-04-05 Final Report, 1994.