Towards High Performance Discrete-Event Simulations of Smart Electric Grids

Kalyan S. Perumalla, James J. Nutaro, and Srikanth B. Yoginath
Oak Ridge National Laboratory, Oak Ridge, TN 37831-6085, USA

ABSTRACT

Future electric grid technology is envisioned on the notion of a smart grid in which responsive end-user devices play an integral part of the transmission and distribution control systems. Detailed simulation is often the primary choice in analyzing small network designs, and the only choice in analyzing large-scale electric network designs. Here, we identify and articulate the high-performance computing needs underlying high-resolution discrete event simulation of smart electric grid operation large network scenarios such as the entire Eastern Interconnect. We focus on the simulator's most computationally intensive operation, namely, the dynamic numerical solution for the electric grid state, for both time-integration as well as event-detection. We explore solution approaches using general-purpose dense and sparse solvers, and propose a scalable solver specialized for the sparse structures of actual electric networks. Based on experiments with an implementation in the THYME simulator, we identify performance issues and possible solution approaches for smart grid experimentation in the large.

Categories and Subject Descriptors

I.6 [**Computing Methodologies**]: Simulation and Modeling; J.2 [**Computer Applications**]: Physical Sciences and Engineering; J.7 [**Computer Applications**]: Computers in Other Systems

General Terms

Algorithms, Design, Experimentation, Performance

Keywords

Discrete Event, Smart Grid, Sparse Solver, Parallel Computing

1. INTRODUCTION

The simulation of hybrid models will have a leading role in the design of control systems for a smart power grid in the large. Historically, industries adopting highly automated systems have motivated and promoted research in hybrid models and their simulators; manufacturing and aerospace are prominent examples. The distributed, automatic controls that are intrinsic to a smart grid will be the motivator for another wave of research in modeling and simulation of hybrid systems, and the size and complexity of electrical power systems will ensure that high performance computing is a part of this future.

Hybrid models emerge from the study of interactions between a system's digital and analog components. The continuous dynamics of analog components are modeled with differential-algebraic equations. Discrete event models are used to describe the dynamics of digital components. The interaction of these discrete event and differential-algebraic models plays a central part in simulations of the complete system.

A simulation of frequency regulation by loads provides one example of a model containing continuous and discrete event components (see, e.g., [9, 10, 13]). Consider in particular a model used to select design requirements for the sensors. At each load, a digital controller watches the frequency of the power system where it is installed. The sensor in the controller has finite precision, and so acts upon only discrete changes in frequency; a typical sensitivity is $\Delta f = 0.005$ Hz (see, e.g., [14]). When the sensor reports a change in frequency, the controller changes the impedance of the load (e.g., by turning electrical equipment on or off) at its location. In the time between these control events, the electromechanical aspects of the power system evolve continuously as described by their differential-algebraic model. One variable in this model is the frequency observed by the sensors. Simulation of the interacting dynamics of these two models – one discrete event and the other continuous – is necessary to predict the effectiveness of the control scheme.

Discontinuity locking is a technique central to the discrete event simulation of hybrid systems (see, e.g., [3]). With this approach, interaction between the discrete event and continuous model occurs at the roots of state event functions. In the example above, the continuous model contains a variable f_k that is the frequency at the k^{th} bus; the sensor has a variable n_k that is the frequency level at which the most recent control event took place at the k^{th} bus; and the next discrete event at the k^{th} bus happens when $f_k - (n_k \pm 1)\Delta f = 0$. This function is an example of a state event function; by construction, it is zero at the event and changes sign upon crossing zero.

During the simulation of the differential algebraic model, all of the discrete variables are used at their present values. In this case, the n_ks are kept constant while simulating the electro-mechanical dynamics. However, at each step of the numerical scheme that solves the differential-algebraic equations, the values of the state event functions are calculated. If the sign of any these functions has changed, then the simulator missed an event (or, in general, at least one event). It then looks with a root-finding procedure for the precise location in time where the state event function changed sign.

At best, the location of the event is found in just one extra evaluation of the differential-algebraic equations. More often, several solutions to these equations must be calculated as the root finding procedure narrows the time interval containing the event. When the event has been found, the discrete event is applied – in our example, the admittance at the bus is modified – and the numerical algorithm restarted at the time of the event.

This discourse highlights three aspects of discontinuity locking that contribute to a high computational cost for simulating hybrid models. The first is that the root finding procedure necessitates a large number of evaluations of the differential-algebraic equations. Secondly, the frequency of events places a potentially severe constraint on the step size of the numerical scheme that is used to simulate the continuous model; this also contributes to a very large number of evaluations of the differential-algebraic systems.

Third, frequent discontinuities caused by discrete events prohibit in practice the use of multi-step numerical methods. Although a number of multi-step numerical methods are available for optimized simulation of electric networks, they cannot be used for discrete event execution, owing fundamentally to their closed-system treatment and/or non-interruptability at irregular points along the system trajectory. While ability of multi-step methods to reuse calculations from previous time steps is useful to improve the accuracy of the numerical solution in next steps, they cannot be used to advance the simulation along irregular time steps. Instead, single step methods that are easy to restart are preferrable; for example, those in the Runge Kutta family. These require multiple evaluations of the differential-algebraic equations to calculate a single point in their trajectory; once again, this increases the number of evalulations of the differential algebraic equations.

To examine wide area control of electro-mechanical transients by smart devices, simulation of electro-mechanical dynamics at the transmission level are essential. Though previous work in power system simulation has addressed this computational problem in a classical setting (the literature is large, but see, e.g., [8, 7, 12] for a glimpse of recent developments), the introduction of significant discrete event dynamics necessitates a new approach.

With regard to the differential algebraic equations that model the electro-mechanical dynamics, it is the solution of the linear system that relates voltages and currents of the transmission network that poses the greatest computational challenge. In particular, we must address two problems: (1) the frequent refactorization of the admittance matrix as required to model some types of control events, and (2) the large numbers of back-substititions imposed by discontinuity locking.

In this paper we present work towards a new algorithm for solving this linear systems problem in the context of discrete event simulation. The new algorithm fits neatly into the simulation framework and will reduce execution times for large-scale models from tens of hours to the few minutes that are needed for a practical design tool. By large-scale, we mean models with thousands to tens of thousands of buses, each with its own discrete sensors and actuators.

2. DISCRETE EVENT SIMULATION

Algorithm 1 shows the discrete event-based execution loop comprising the computational load of the simulation. It proceeds in varying increments δt_{min} of simulation time, the increments being determined by the state of the system, the desired integration error limits, the desired accuracy of smart grid device operations, and any user-specified phenomena scheduled as events to occur along the simulation time.

The bulk of the computational burden of the simulation arises from the computation at Step 3(a)-(c), which is at the heart of the discrete event-based approach, namely, determination of a safest/correct leap in simulation time permissible by the current state of the system. Every increment of simulation time involves computing the solutions for voltages in the linear problem given by the matrix equation $YV = I$, where Y is an $n \times n$ matrix obtained from the admittance relating the buses via line dependencies, V is the vector of voltages at each bus, and I is the vector of injected currents at the buses. The matrix equation is solved with multiple right-hand side vectors, one for every integration step trial and for threshold crossing time trial. The matrix Y is factored on demand, whenever any of its elements are modified in the simulation loop, and the factors are saved unless and until the matrix changes again. The saved matrix factors are used to solve for voltages for each right hand side. The matrix may be modified either by the smart device operations at Step 5, or by externally modeled effects at Step 6.

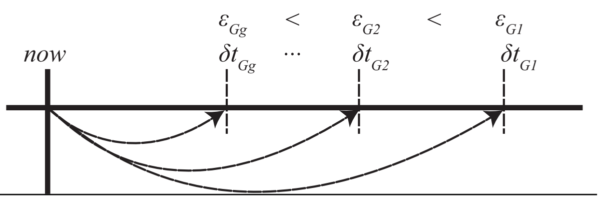
Within each i^{th} time increment (each iteration of the simulation loop), let G_i be the number of integration step trials, H_i be the number of threshold-crossing events tested, and E_i be the number of scheduled external events. Every integration step trial requires a matrix solution, as does every check for threshold-crossing. Let F_i be the number of times matrix factorization is performed per iteration, and S_i be the number of matrix solutions performed per integration or threshold-crossing check operation in each iteration. One additional solve is needed if an external event induces a change in the network. Let $\eta = 1$ if an external event results in a change in the matrix, and $\eta = 0$ otherwise. Then, the total simulation time is dominated by the time for matrix operations $T_{matrix} = \sum_i (G_i + H_i + \eta) \times S_i \times T_S + F_i \times T_F$, where T_S is the time for one matrix solution, and T_F is the time for one matrix factorization. Typically, $F_i \leq 1$, although one can construct scenarios in which $F_i > 1$.

3. IMPLEMENTATION

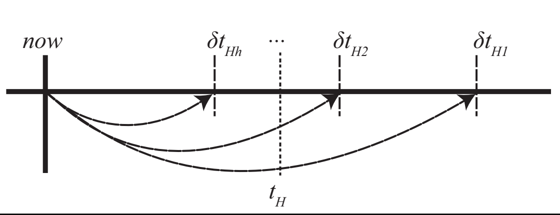
The THYME electric grid simulator, developed by Nutaro et al [11], is capable of modeling transmission and generation with high fidelity. It is written in C^{++}, portable to several platforms, and uses the Message Passing Interface (MPI) for parallel execution. The simulator accepts grid networks in standard formats such as the IEEE CDF format, and adds sensor behavior as C^{++} class methods in a pre-defined class hierarchy that can be easily customized for a variety of smart device behaviors. The classes subscribe to

Algorithm 1 Discrete event-based simulation execution

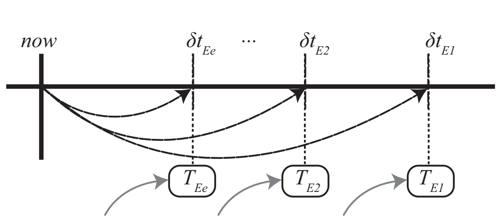
```
1: $now \leftarrow 0$
2: while $now < endtime$ do
3:     $\delta t_{min} \leftarrow \min(\delta t_{integrator}, \delta t_{threshold}, \delta t_{external})$
       where, the $\delta t$s are determined as follows:
         (a)  $\delta t_{integrator} \leftarrow \delta t_{Gg}$ for the largest time leap, $\delta t_{Gg}$, that
              gives an acceptable error $\varepsilon_{Gg}$ in numerical integration
              of the grid system state from $now$ to $now + \delta t_{integrator}$
```


ε_{Gg} < ε_{G2} < ε_{G1}
now
δt_{Gg} ··· δt_{G2}
δt_{G1}

```
         (b)  $\delta t_{threshold} \leftarrow \delta t_{Hh}$ for the earliest time leap $\delta t_{Hh}$ near
              the earliest time $t_H$ at which the system state (e.g.,
              voltage) crosses a threshold value specified by a model
              component (e.g., automated control from smart de-
              vices for voltage shedding)
```



```
         (c)  $\delta t_{external} = T_E - now$ for $T_E = \min_i(T_{Ei})$, which is
              the earliest of all times $\{T_{Ei}\}$ at which an externally-
              specified system change is scheduled to occur (e.g.,
              outages due to non-electrical causes)
```



```
4:     Advance the electric grid state from $now$ by $\delta t_{min}$
5:     Incorporate electric device control effects in the interval $now$ and $now + \delta t_{min}$
6:     Incorporate effects of external events, if any exist with time stamp $\leq now + \delta t_{min}$
7:     Advance the simulation time: $now \leftarrow now + \delta t_{min}$
8: end while
```

events raised by the simulator for changes of interest such as voltage/frequency threshold crossings. While the core transmission network can be specified with input files, additional detail can be introduced into the network by adding subnetworks as surrogates for distribution networks attached to user-specified transmission nodes. The synthetic distribution networks are generatable at simulator initialization using Algorithm 2. Table 1 shows sample simulation output from experiments of a generator failure executed within the discrete event framework on different networks.

Parallel execution is realized in a master-worker framework using MPI. Rank 0 hosts the simulation loop, while ranks > 0 run the slave loop in which they receive commands from the master to participate in parallel matrix solutions at appropriate points in simulation.

As the structure of the admittance matrices reflects the connectivity of the actual, sparsely-connected electric grid, the admittance matrices are found to be extremely sparse. Figure 2 shows the degree distributions observed on sample grids. While dense solvers can be used for smaller networks (e.g., those with less than a few hundred buses), sparse solvers become imperative on larger networks to minimize factorization times and solution times. On networks such as the IEEE 118 and IEEE 300, implementation of the basic linear algebra services (BLAS) [2] was found to be adequate to deliver competitive run times relative to sparse solvers. For larger networks, such as the ERCOT and the Eastern Interconnect cases, sparse solvers such as SuperLU [5] or MUMPS [1] factor and solve the matrices faster. However, even with sparse matrix operations, the total computation time remains very high for any large scenario. Parallel execution is the only recourse to reduce the run time.

The simulator is structured such that different solvers can be incorporated in a plug-and-play fashion for the discrete event detection. Solvers that are incorporated include BLAS, SuperLU, MUMPS, and Blocktri (described later).

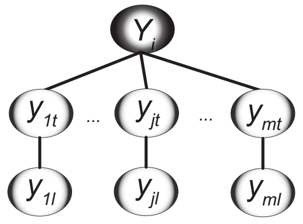
4. RUNTIME PERFORMANCE

The performance of the simulator has been tested with two different simulation scenarios on four different grid network topologies. In the first scenario, an outage of a generator is simulated; the network stablizes after brief frequency excursions due to the loss of the generator. In the second scenario, sensor devices are attached at every bus which subscribe to simulator events that detect voltage variation beyond sensor-specified thresholds.

The time advances observed in the first scenario varied from 10^{-2} to 10^{-1} seconds, with the smaller advances incurred during the generator-outage transients. Time advances in the second scenario varied from 10^{-4} to 10^{-3} seconds, due to the tight thresholds established by the sensors, and due to the large number of staggered states induced by the geographically distributed sensors in the scenario. The number of matrix factorization operations also corre-

Algorithm 2 Synthetic network expansion from transmission to distribution

```
1: for node $n_i$, $0 \le n_i < N$ where $N$ is the number of nodes do
2:     $Y_i \equiv$ inductance at node $n_i$
3:     $m \equiv$ number of branches (subsystems) in distribution attached to node $n_i$
4:     $r \equiv$ desired ratio $\frac{y_{it}}{y_{il}}$
5:     Expand the network as follows:
```



```
6:     $y_{remaining} \leftarrow Y_i$
7:     for $j = 0$ to $m - 1$ do
8:         if $j < m - 1$ then
9:             $y_j \leftarrow$ uniform random number in $\frac{y_{remaining}}{m-j} \pm c\%$ (for some desired value of $c$)
10:        else
11:            $y_j \leftarrow y_{remaining}$
12:        end if
13:        $y_{jl} \leftarrow y_j \times \frac{(r+1)}{r}$
14:        $y_{jt} \leftarrow r \times y_{jl}$
15:        Add a new node (number of nodes now increases to $N + i$)
16:        Add a new line ($N + i$, $N + i$) with admittance $y_{jl}$
17:        Add a new line ($n_i$, $N + i$) with admittance $y_{jt}$
18:        $y_{remaining} \leftarrow y_{remaining} - y_j$
19:    end for
20: end for
Ensure: The network now contains $m \times N$ nodes
```

spondingly increased in the second scenario. The number of matrix solution operations per factorization operation also increased.

Figure 1 shows the runtime performance obtained by executing the generator failure scenario on the ERCOT network. The simulation is performed in parallel on varying number of processor cores of a Cray XT4 machine in which each node has four processor cores (1 quad-core AMD Budapest) and 8GB of main memory. Each experiment is executed with two settings:

- **MT**: multi-threading is enabled, with four threads per node, and one MPI task per node, and
- **No-MT**: multi-threading disabled, with four MPI tasks per node, i.e., one task per core.

The measures of interest are the number of factorization and solution operations, their runtime costs, and the total runtime of the simulation.

The top part of Figure 1 shows the time per factor and time per solve, each further qualified by **MT** and **No-MT**. It is observed that the time to factor decreases by increasing the number of processor cores when multi-threading is used. However, the non-multi-threaded execution runs faster than the multithreaded version, but does not scale with parallel execution (in fact, it suffers increased costs at 64 processor cores). Similar trend is observed for the time per solve operation. The solve operation, interestingly, executes an order of magnitude faster than the factor, even though the matrix is highly sparse. However, in the generator failure scenario, the number of factor operations is observed to be far fewer than the number of solves, which accentuates the otherwise lower cost per solve. This is observed in the middle portion of Figure 1 which shows the total time spent in factors and the total time spent in solves. The total solve time dominates the total simulation time, which is counter-intuitive to the expectation that factorization time $\mathcal{O}(N^3)$ is much larger than that of solution time $\mathcal{O}(N^2)$. This is because of the significant number of distinct right hand side vectors that need to be solved on average per factor during the discrete event execution, specifically, in Step 3(a)-(c). The number of solves per factor is in fact observed to be over 200, as shown in the bottom part of Figure 1.

5. SCALABLE SOLVER

Based on runtime experiments, we found that the general-purpose sparse solvers do not exhibit the required scalability with the number of processors. This can be attributed to the nature of the admittance matrices which are not only extremely sparse but also highly irregular in their non-zero pattern. A solver that can exploit the special structure of the networks can help improve scalability. We identify here a provably scalable method to perform this, and apply it to the electric grid networks. This approach relies on minimizing the span of non-zeros from the diagonal, which is characterized by a well-known metric called bandwidth, elaborated next.

5.1 Admittance Matrix Bandwidth

The bandwidth of a matrix is the farthest distance of any non-zero element from the diagonal. The smaller the bandwidth, the better the potential for efficient parallel matrix operations. While the problem of finding the specific permutation that minimizes the bandwidth is an NP-complete problem, heuristics can be used to reduce the bandwidth. We used a variant of the Cuthill-Mckee method [4] for bandwidth minimization, and found that the bandwidth can be dramatically reduced, relative to the bandwidth of the networks as given in the standard input data sets. The reductions thus obtained are illustrated in Table 2, for the four data sets used here: IEEE 118, IEEE 300, ERCOT, and EI_NERC_09s.

Since the elements of the admittance matrix change during the course of the simulation, the bandwidth minimization operation must be performed dynamically before every factorization operation. This implies that the bandwidth minimization algorithm itself must be implemented and incorporated efficiently into the main discrete event loop of the simulation.

In the larger networks (ERCOT and Eastern Interconnect EI_NERC09s), since the matrix bandwidth is observed to be relatively small compared to the total matrix size, it makes it possible to conceive of a provably scalable algorithm that can solve a bandwidth-minimized matrix, as discussed next.

5.2 Block-Diagonal Solver

Given a grid with n nodes, the admittance matrix A of $n \times n$ complex numbers is determined. Using fast heuristics, the nodes are permuted such that matrix bandwidth M is as small as possible. The matrix can then be viewed as a block

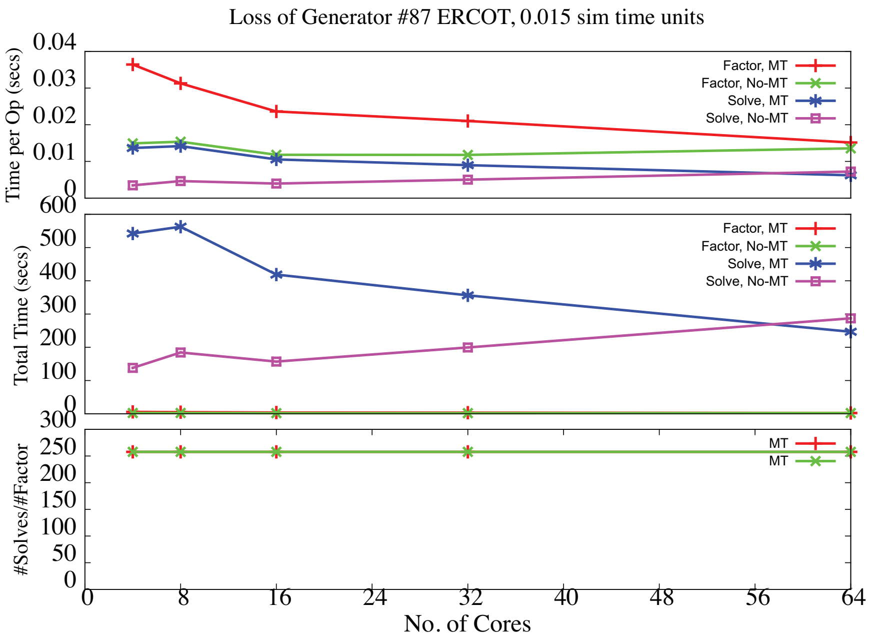


Figure 1: Discrete event simulation runtime performance on a Cray XT4

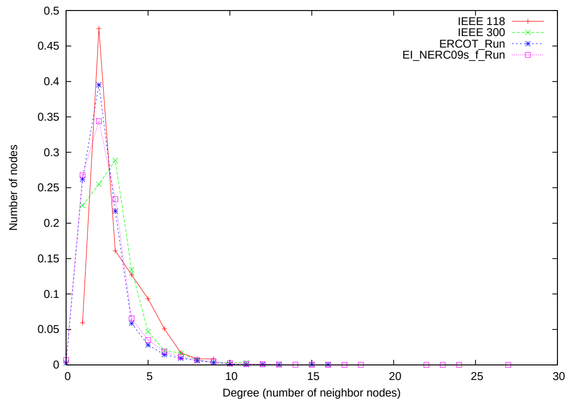


Figure 2: Degree distributions of sample grid networks

tri-diagonal matrix of N blocks, each block of size $M \times M$, where $N = \lceil \frac{n}{M} \rceil$. Its diagonal blocks $D_i, 1 \leq i \leq N$ are matrices, each of size $M \times M$. Its lower diagonal blocks $L_i, 1 < i \leq N$ are upper-triangular dense matrices, and its upper diagonal blocks $U_i, 1 \leq i \leq N-1$ are lower-triangular matrices. The last $s = (NM - n)$ rows of L_N, D_N and U_N are padded appropriately in case $s > 0$, by setting the last s rows of L_N, D_N and U_N to be all zeroes except for the last s diagonal entries of D_N to be identity.

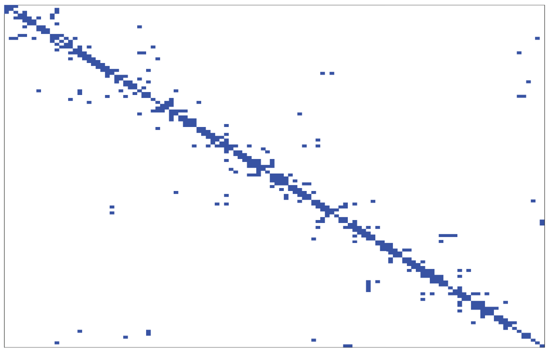
A block-cyclic reduction-based scheme can be utilized to solve such a block-tridiagonal matrix in a provably scalable time of $\mathcal{O}(\log n)$. A similar scheme has recently been used successfully in other scientific codes such as plasma physics[6]. A major difference, however, is that the tridiagonal matrices of the earlier works contained blocks that are dense sub-matrices, but the blocks in the bandwidth-minimized admittance matrices are observed to be extremely sparse. This difference makes it necessary to utilize sparse solves as *sub-solvers* inside the block cyclic solvers of the overall matrix. As seen in Table 2, in the largest case of the Easter Interconnect network (EI_NERC09s), $n = 45,552$, and the minimized bandwidth $M = 1600$. Since the block size of 1600 is too large for dense sub-matrix operations, we observe degraded efficiency when a dense solver (BLAS) is used for the blocks in the block-cyclic reduction. It is clear that the block cyclic reduction, while providing scalability, remains to be enhanced with efficiency by incorporating a sparse solver *within* the recursive reduction algorithm of the overall matrix.

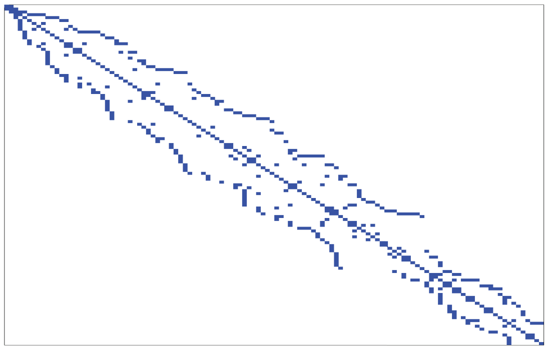
The complete scheme for the solution is illustrated in Figure 4, starting from a given admittance matrix to be solved, to the permutation for bandwidth minimization, and factorization, and then to the actual solution for any given right hand side.

We are currently implementing the sparse solution using MUMPS within BLOCKTRI; in fact, multiple simultaneously active MUMPS instances are invoked at the same time during the same parallel run of a single factorization/solve operation via BLOCKTRI. Additional work is pending in executing the networks under this solver, and observing its scalability to even larger networks, and with futuristic automated controls based on ubiquitous sensors.

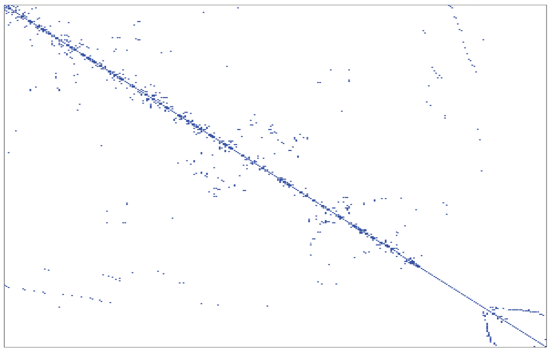
6. SUMMARY AND FUTURE WORK

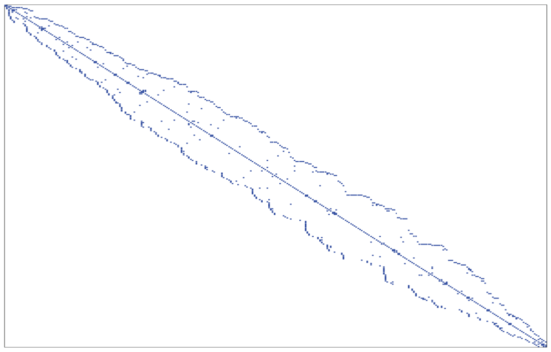
Experimentation for the designs of futuristic smart grids at regional to national scale require the use of discrete event style of simulations for capturing the dynamics at the right levels of fidelity. However, this style of execution presents unique core computational requirements that are relatively different from traditional electric grid simulators. Our efforts are aimed at filling this need, by developing the algorithms and implementations in parallel software. Preliminary experiments show the feasibility of the discrete event


Input (bandwidth 105)

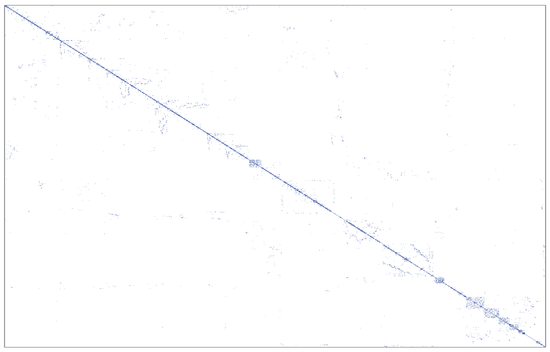

Permuted (bandwidth 18)

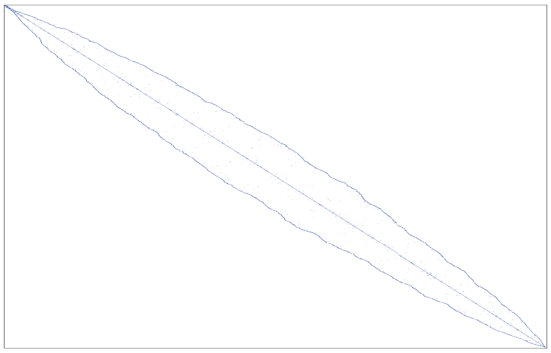
IEEE 118 case, with 118 nodes and 358 buses


Input (bandwidth 244)

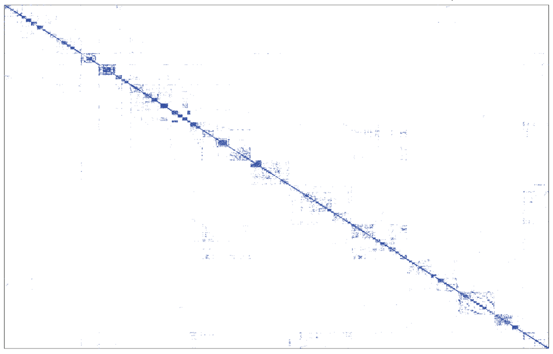

Permuted (bandwidth 36)

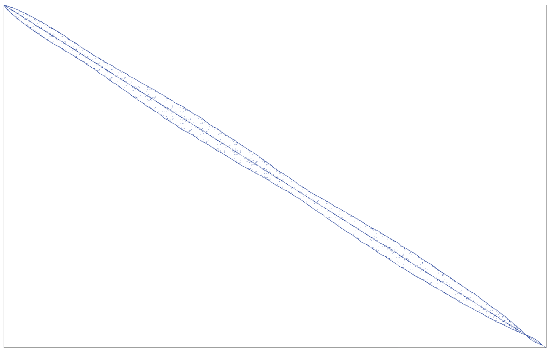
IEEE 300 case, with 298 nodes and 814 buses


Input (bandwidth 4434)


Permuted (bandwidth 591)

ERCOT case, with 5357 nodes and 12572 buses


Input (bandwidth 38713)


Permuted (bandwidth 1600)

EI_NERC09s_f case, with 45,552 nodes and 111,576 buses

Table 2: Original and bandwidth-optimized admittance matrices

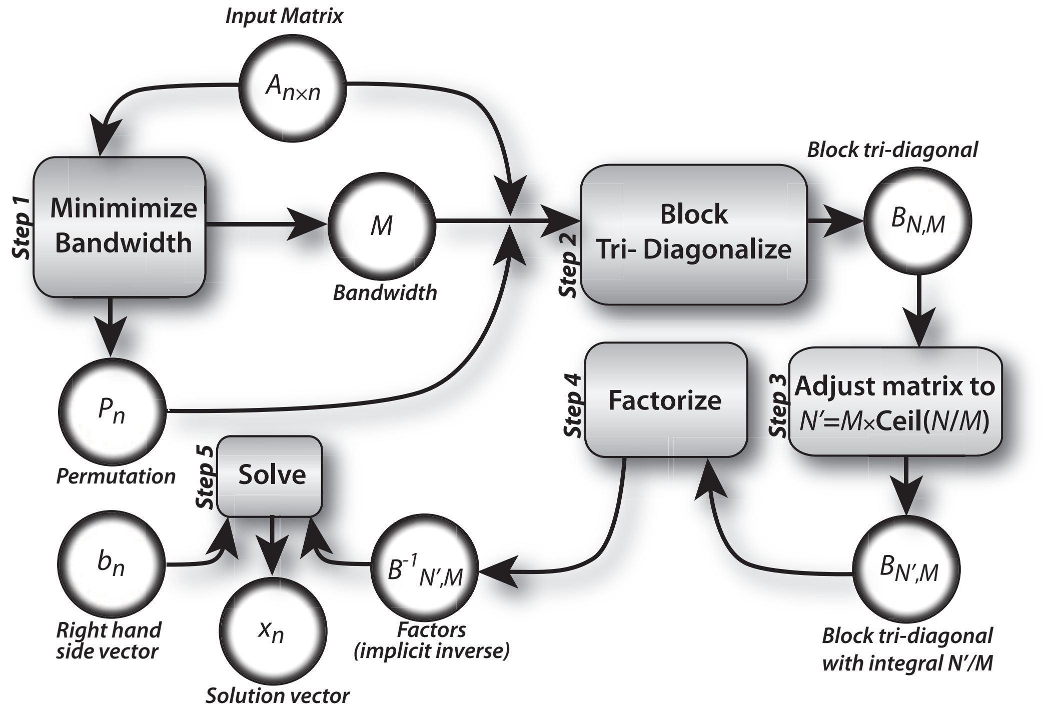


Figure 4: Solution algorithm to solve a sparse admittance matrix A of dimension $n \times n$.

approach in accurately capturing electro-mechanical dynamics in a manner that is suitable to attach complex, smart device behaviors for future device/protocol design and experiments. The runtime performance studies also clearly show that the most significant computational burden lies in the solution of the admittance matrices at various points of event execution. We identified a provably scalable solution approach to solving very large electric grid network scenarios with several thousands of buses. The implementation of this customized solver is underway, and its scalability and efficiency characteristics are ongoing work. Future work includes addressing the validation and cross-checking needs, as well as collaborations with other researchers in the visualization of geographically distributed phenomena in large-scale smart grids.

Acknowledgements

This paper has been authored by UT-Battelle, LLC, under contract DE-AC05-00OR22725 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 manuscript, or allow others to do so, for United States Government purposes. This effort has been supported by research sponsored by the Laboratory Directed Research and Development Program of Oak Ridge National Laboratory. The research used resources of the National Center for Computational Sciences (NCCS) at Oak Ridge National Laboratory, which is supported by the Office of Science of the Department of Energy.

7. REFERENCES

[1] P. R. Amestoy, I. S. Duff, J. Koster, and J.-Y. L'Excellent. A fully asynchronous multifrontal solver using distributed dynamic scheduling. *SIAM Journal on Matrix Analysis and Applications*, 23(1):15–41, 2001.

[2] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. D. Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, third edition, 1999.

[3] F. E. Cellier and E. Kofman. *Continuous system simulation*. Springer, 2006.

[4] E. Cuthill and J. McKee. Reducing the bandwidth of sparse symmetric matrices. In *Proceedings of the 1969 24th national conference*, ACM '69, pages 157–172, New York, NY, USA, 1969. ACM.

[5] J. W. Demmel, S. C. Eisenstat, J. R. Gilbert, X. S. Li, and J. W. H. Liu. A supernodal approach to sparse partial pivoting. *SIAM J. Matrix Analysis and Applications*, 20(3):720–755, 1999.

[6] S. Hirshman, K. Perumalla, V. Lynch, and R. Sanchez. Bcyclic: A parallel block tri-diagonal matrix cyclic solver. *Journal of Computational Physics*, 229:6392–6404, 2010.

[7] V. Jalili-Marandi and V. Dinavahi. Simd-based large-scale transient stability simulation on the graphics processing unit. *IEEE Transactions on Power Systems*, 25(3):1589 –1599, 2010.

[8] S. Jin, Z. Huang, Y. Chen, D. G. Chavarria-Miranda, J. Feo, and P. C. Wong. A novel application of parallel betweenness centrality to power grid contingency

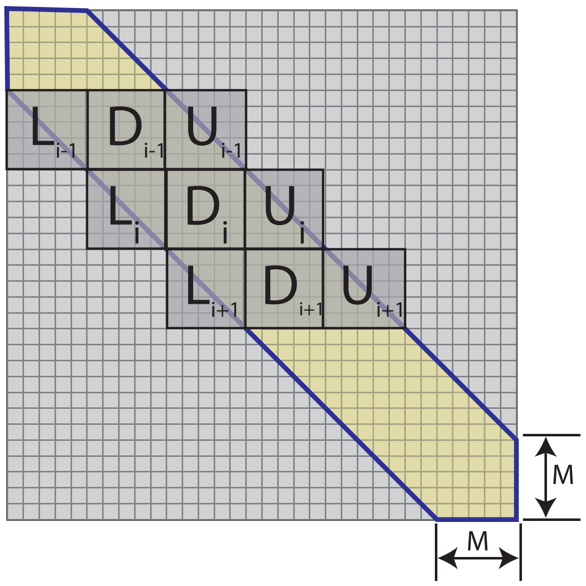


Figure 3: Matrix of bandwidth M viewed as a block-tridiagonal matrix of block size M, with lower diagonal blocks L_i being upper-triangular, and upper diagonal blocks U_i being lower-triangular

analysis. In *International Parallel and Distributed Processing Symposium/International Parallel Processing Symposium*, pages 1–7, 2010.

[9] A. Molina-GarciÌĄ, F. Bouffard, and D. Kirschen. Decentralized demand-side contribution to primary frequency control. *IEEE Transactions on Power Systems*, 26(1):411 –419, feb. 2011.

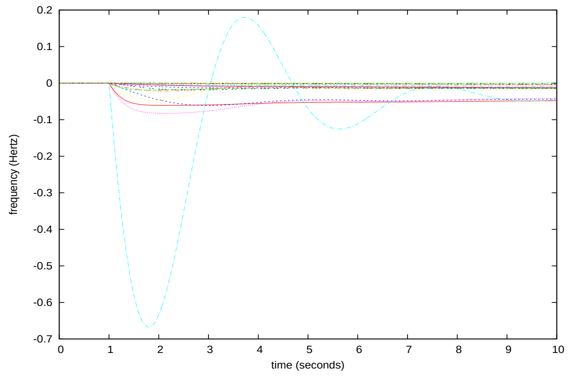
[10] S. Mullen and G. Onsongo. Decentralized agent-based underfrequency load shedding. *Integrated Computer-Aided Engineering*, 17(4):321–329, 2010.

[11] J. Nutaro, P. T. Kuruganti, V. Protopopescu, and M. Shankar. The split system approach to managing time in simulations of hybrid systems having continuous and discrete event components. *SIMULATION*, May 2011.

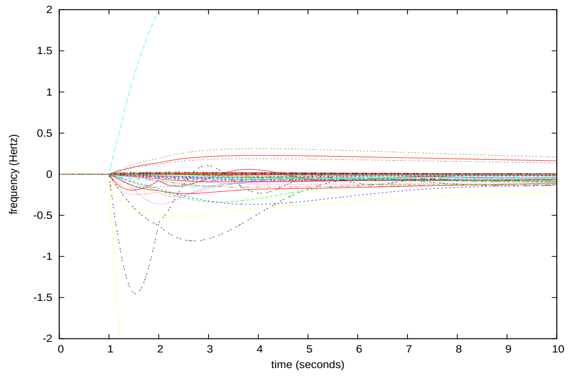
[12] J. Shu, W. Xue, and W. Zheng. A parallel transient stability simulation for power systems. *IEEE Transactions on Power Systems*, 20(4):1709 – 1717, 2005.

[13] D. Trudnowski, M. Donnelly, and E. Lightner. Power-system frequency and stability control using decentralized intelligent loads. In *Proceedings of the 2005 IEEE Power Engineering Society T & D Conference and Expo*, pages 1453–1459, 2006.

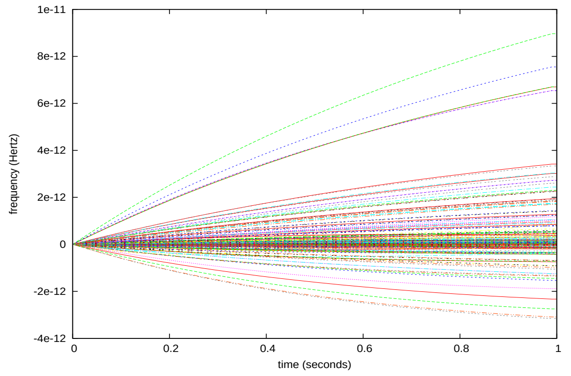
[14] S.-J. Tsai, L. Zhang, A. Phadke, Y. Liu, M. Ingram, S. Bell, I. Grant, D. Bradshaw, D. Lubkeman, and L. Tang. Frequency sensitivity and electromechanical propagation simulation study in large power systems. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 54(8):1819–1828, 2007.



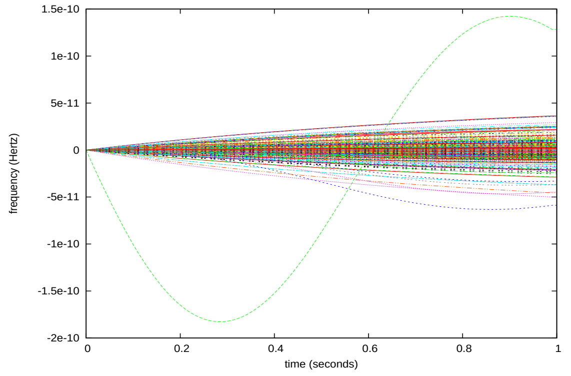
IEEE 118



IEEE 300



ERCOT



EI_NERC09s_f

Table 1: Sample simulation output showing the transient frequency excursions due to the outage of a single generator (timescales are different across networks)