Parallel Dynamics Simulation Using a Krylov-Schwarz Linear Solution Scheme

Shrirang Abhyankar Member, IEEE, Emil Constantinescu, Barry F. Smith, Alexander J. Flueck Senior Member, IEEE, Daniel A. Maldonado

Abstract—Fast dynamics simulation of large-scale power systems is a computational challenge because of the need to solve a large set of stiff, nonlinear differential-algebraic equations at every time step. The main bottleneck in dynamic simulations is the solution of a linear system during each nonlinear iteration of Newton’s method. In this paper, we present a parallel Krylov-Schwarz linear solution scheme that uses the Krylov subspace-based iterative linear solver GMRES with an overlapping restricted additive Schwarz preconditioner. Performance tests of the proposed Krylov-Schwarz scheme for several large test cases ranging from 2,000 to 20,000 buses, including a real utility network, show good scalability on different computing architectures.

Index Terms—Dynamics simulation, GMRES, iterative linear solution, Krylov subspace, Parallel processing, Schwarz preconditioning, Transient stability, Power system dynamics

I. INTRODUCTION

Dynamics simulation, or transient stability analysis, is an important function in the planning, design, and post-disturbance analysis of power systems but its application is restricted to offline studies because of the computational requirements. Providing system operators with the capability to examine the system dynamics in real-time, or faster than real time, creates a new paradigm in power system operation by facilitating real time or predictive control actions. This is particularly important because the power system behavior is becoming increasingly dynamic.

A number of factors contribute to this increased dynamic behavior of power systems. First, today’s power systems are being stressed due to increasing load in certain zones, yet transmission line capacity is stagnant. Second, the economic benefits of deregulation have pushed the system operation to its limits, decreasing the stability margins. Third, the distribution system is becoming increasingly “active”, leading to possible two-way power flows from distribution to transmission. Fourth, the lowering of system inertia, as more renewable energy sources are installed, has introduced frequency and voltage control problems. Fifth, situational awareness of variability and unpredictability of these renewable energy sources, for example, decreased output of a solar PV installation due to cloud cover, is a need expressed by the operators. Finally, the prediction and mitigation of cascading outages, that cannot be predicted by snapshots of the current operating state obtained every few seconds by state estimation, is an important concern for utilities.

This material is based upon support by U.S. Dept. of Energy, Office of Electricity, Advanced Grid Modeling Research, under Contract DE-OE0000624

A key requirement to achieve real-time dynamics analysis is the acceleration of dynamics simulation tools. This need for faster power grid dynamics simulation (or transient stability analysis) has also been expressed by the power system community in recent years [1], [2]. However, real-time dynamics simulation of a utility-sized networks comprising thousands of buses is challenging because of the difficulty in solving a set of equations containing tens of thousands of variables in a few milliseconds. For instance, Huang and Nieplocha [3] reported that a simulation of 30 seconds of dynamic behavior of the Western Interconnection requires about 10 minutes of computation time today on an optimized single processor. As processor speeds were increasing, real-time dynamics simulation appeared possible in the not-too-distant future. Unfortunately, processor clock speeds saturated about a decade ago, and real-time dynamics simulation remains a grand computing challenge.

Dynamics simulation of a large-scale power system is computationally challenging because of the presence of a large set of stiff, nonlinear differential-algebraic equations (DAEs), where the differential equations model dynamics of the rotating machines (e.g., generators and motors) and the algebraic equations represent the transmission system and quasi-static loads. The electrical power system is expressed as a set of nonlinear DAEs, as given in (1), where \( f : \mathbb{R}^m \times \mathbb{R}^n \rightarrow \mathbb{R}^m \) and \( g : \mathbb{R}^m \times \mathbb{R}^n \rightarrow \mathbb{R}^n \) are vector-valued nonlinear functions that model the quasi steady-state dynamics of the rotating machines and the network, respectively, and \( x \in \mathbb{R}^m \) and \( y \in \mathbb{R}^n \) are the dynamic states of the machines and the network, respectively.

\[
\frac{dx}{dt} = f(x, y) \\
0 = g(x, y)
\]  

The solution of the dynamic model given in (1) using a simultaneous implicit (SI) solution scheme [4] needs the following:

- A numerical integration scheme, for example Implicit trapezoidal, to convert the differential-algebraic equations to purely algebraic form
- A nonlinear solution scheme (typically Newton’s method) to solve the resultant nonlinear algebraic equations
- A linear solver to solve the update step (direction) at each iteration of the nonlinear solution

Prior experience in accelerating dynamics simulations of large power systems [5], [6] revealed that the main bottleneck in these simulations is the solution of the linear system during
each iteration of Newton’s method, typically constituting 90-95% of the total execution time.

In this paper, we present parallel dynamics simulation accelerated by a combination of a Krylov subspace iterative linear solver and a Schwarz preconditioner. Specifically, the Krylov-subspace iterative solver GMRES (Generalized Minimum Residual Method) is used with restricted overlapping additive Schwarz preconditioning. This approach was found to be scalable for the solution of large power flow problems in previous work [7]. This paper presents a detailed evaluation of the proposed linear solution scheme on four test systems, including a real utility network, on two different computer architectures. Test results on these dynamics cases demonstrate good scalability of the proposed approach in comparison with using a parallel direct linear solver.

The organization of this paper is as follows. Section II presents an overview of parallel dynamics simulation approaches. Domain decomposition of the power system dynamics equations (1) is discussed in section III. Sections IV and V present the details of the Krlov-subspace iterative linear solution scheme and Schwarz preconditioning, respectively. Section VI presents the test system and multicore machine details and discusses the performance of the proposed approach on different test systems. Section VII summarizes our conclusions.

II. PARALLEL DYNAMICS SIMULATIONS

A natural way to speed up simulations is to use parallel computing techniques, namely, share the computational load among multiple processing units. In the context of parallel algorithms for dynamics simulations, most of the research effort was done over the past decade. Three parallelization approaches - spatial parallelization, temporal parallelization, waveform relaxation (hybrid space-time parallelization) - have been explored by power system researchers.

The parallel-in-space algorithms partition the given network into loosely coupled or independent subnetworks. Each processor is then assigned equations for a subnetwork. The partitioning strategy for the network division is critical for parallel-in-space algorithms to minimize the coupling between the subnetworks, that is, to reduce the interprocessor communication, and balance the work load. Once a suitable partitioning strategy is selected, the next step is the solution of the linear system in each Newton iteration. Several linear solution schemes have been proposed in the literature, of which the most prominent are the very dishonest Newton method and the conjugate gradient method. Chai et. al., [8] use the very dishonest Newton method in which the factorization of the Jacobian matrix is done only when a certain fixed number of iterations are exceeded. Decker et. al., [9] decomposes the network equations in a block-bordered diagonal form and then use a hybrid solution scheme using LU and conjugate gradient. In [10], Decker et. al. solve the network by a block-parallel version of the preconditioned conjugate gradient method. The network matrix in [10] is in a near block diagonal form. A parallel Schur-complement-based domain decomposition approach was investigated in [11] for accelerating the dynamics simulation on shared-memory machines.

The parallel-in-time approach for dynamics simulations was introduced in [12]. The idea is to combine the differential and algebraic equations over several time steps, create a bigger system, and solve them simultaneously by using the Newton method. All the equations for several integration steps are assigned to each processor.

Waveform relaxation methods [13], [14], [15] involve a hybrid scheme of space and time parallelization in which the network is partitioned in space into subsystems and then distributed to the processors. Several integration steps for each subsystem are solved independently to get a first approximation [16]. The results are then exchanged, and the process is repeated. The advantage of this scheme is that each subsystem can use a different integration step and/or different integration algorithm (multirate integration).

III. DOMAIN DECOMPOSITION

Domain decomposition algorithms operate by dividing the entire domain $W$ into smaller non-overlapping subdomains $W_i^0$ where

$$W = \bigcup_{i=1}^{N} W_i^0$$

and then assigning each subdomain $W_i^0$ to a processing unit. In the context of power systems, this approach means partitioning the power system network into several subnetworks where each subnetwork is assigned to a processing unit. Figure 1 shows an example of the division of the IEEE 118-bus system into two subnetworks. Each subnetwork is then the domain of operation of a processing unit; in other words, each processing unit is assigned the DAE for the subnetwork. Equation (3) represents the equations for each processing unit, where the subscript $p$ represents the variables for the current processing unit and the subscript $c$ represents the variables needed from other processing units to compute the function on the current processing unit.

$$\frac{dx_p}{dt} = f(x_p, y_p, y_c)$$

$$0 = g(x_p, y_p, y_c)$$

Note that the differential equations (i.e. the electromechanical machine equations) are naturally decoupled because they
are incident at a bus only, whereas the algebraic network equations require communication with other processing units to compute their local function. Hence the partitioning of the network is done by using only the topology of the network. The partitioning of the network can be done by hand via judicious topology scanning or by graph-partitioning techniques. In this work, we use the ParMetis package [17] for performing the network partitioning.

IV. KRYLOV SUBSPACE ITERATIVE METHODS

Krylov subspace iterative methods are the most popular among the iterative methods for solving large linear systems. These methods are based on projection onto subspaces called Krylov subspaces of the form \( b, Ab, A^2b, A^3b, \ldots \). A general projection method for solving the linear system

\[
Ax = b
\]

is a method that seeks an approximate solution \( x_m \) from an affine subspace \( x_0 + K_m \) of dimension \( m \) by imposing

\[
b - Ax_m \perp L_m,
\]

where \( L_m \) is another subspace of dimension \( m \) and \( x_0 \) is an arbitrary initial guess to the solution. A Krylov subspace method is a method for which the subspace \( K_m \) is the Krylov subspace

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

where \( r_0 = b - Ax_0 \). The different versions of Krylov subspace methods arise from different choices of the subspace \( L_m \) and from the ways in which the system is preconditioned.

A. Generalized Minimum Residual Method

GMRES [18] is a projection method based on taking \( L_m = AK_m(A, r_0) \) in which \( K_m \) is the \( m \)th Krylov subspace. This technique minimizes the residual norm over all vectors \( x \in x_0 + K_m \). In particular, GMRES creates a sequence \( x_m \) that minimizes the norm of the residual at step \( m \) over the \( m \)th Krylov subspace

\[
||b - Ax_m||_2 = \min ||b - Ax||_2.
\]

At step \( m \), an Arnoldi process is applied for the \( m \)th Krylov subspace in order to generate the next basis vector. When the norm of the new basis vector is sufficiently small, GMRES solves the minimization problem

\[
y_m = \arg\min ||\beta e_1 - H_m y||_2,
\]

where \( H_m \) is the \((m+1) \times m\) upper Hessenberg matrix.

B. Preconditioning

The convergence of the Krylov subspace linear solvers depends on the eigenvalues of the operating matrix \( A \) and can be slow if the matrix has widely dispersed eigenvalues, such as ill-conditioned power system matrices. Hence, in order to speed the convergence, a preconditioner matrix \( M \), where \( M^{-1} \) approximates \( A^{-1} \), is generally used. A preconditioner is a matrix that transforms the linear system

\[
Ax = b
\]

into another system with a better spectral properties for the iterative solver. If \( M \) is the preconditioner matrix, then the transformed linear system is

\[
M^{-1}Ax = M^{-1}b.
\]

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

\[
AM^{-1}y = b, \quad x = M^{-1}y,
\]

or split preconditioning

\[
M_1^{-1}AM_2^{-1}y = M_1^{-1}b, \quad x = M_2^{-1}y,
\]

where the preconditioner is \( M = M_1M_2 \).

Designing a good preconditioner depends on the choice of iterative method, problem characteristics, and so forth. In general a good preconditioner should be inexpensive to construct and apply, and the preconditioned system should be easy to solve.

V. OVERLAPPING SCHWARZ PRECONDITIONING

In the area of domain decomposition algorithms, considerable research has been done on overlapping Schwarz algorithms [19], [20], [21], [22]. Our interest for this paper is on an overlapping additive Schwarz preconditioner (ASM), and in particular its variant the restricted additive Schwarz preconditioner introduced by Cai and Sarkis [22], which the authors found to be scalable in a previous work [7]. We first describe the general mechanism of constructing an ASM preconditioner as described in [19] and then present the variant proposed by Cai and Sarkis [22].

A. Additive Schwarz Method (ASM)

Consider a linear system of the form

\[
Ax = b,
\]

where \( A = (a_{ij}) \) is an \( n \times n \) nonsingular sparse matrix, having a symmetric nonzero pattern. Define the graph \( G = (W, E) \), where the set of vertices \( W = \{1, \ldots, n\} \) represent the \( n \) unknowns and the edge set \( E = \{(i, j) | a_{ij} \neq 0\} \) represents the pairs of vertices that are coupled by a nonzero element in \( A \). Here \( n \) is the size of the matrix. Since the nonzero pattern is symmetric, the adjacency graph \( G \) is undirected. Assume that the graph partition has been applied, resulting in \( N \) non-overlapping subsets \( W_i^0 \) whose union is \( W \). Define \( \{W_i^1\} \) as the one overlap partition of \( W \), where \( W_i^1 \supset W_i^0 \) is obtained by including all the immediate neighboring vertices of the vertices in \( W_i^0 \). An example for a domain with two subdomains with a one overlap partition is shown in Figure 2.

Extending this to \( \delta \)-overlaps, we can define the \( \delta \)-overlap partition of \( W \) as

\[
W = \bigcup_{i=1}^{N} W_i^\delta,
\]
where $W_i^\delta \supset W_i^0$ with $\delta$ levels of overlaps with its neighboring subdomains. With each subdomain $W_i^0$, we define a restriction operator $R_i^0$. In matrix terms, $R_i^0$ is a $n \times n$ sub-identity matrix whose diagonal elements are set to one if the corresponding node belongs to $W_i^0$ and to zero otherwise. Similarly for $\delta$-overlapping, the restriction operator $R_i^\delta$ can be defined for each $W_i^\delta$. With this, the subdomain matrix can be defined as

$$A_i = R_i^\delta A R_i^{-\delta}. \quad (10)$$

Note that although $A_i$ is not invertible, its restriction to the subspace

$$A_{i|L_i}^{-1} = (A_{i|L_i})^{-1}$$

can be inverted. Here $L_i$ is the vector space spanned by the set $W_i^\delta$ in $\mathbb{R}^n$. The additive Schwarz preconditioner can be defined as

$$M_{AS}^{-1} = \sum R_i^0 A_{i|L_i}^{-1} R_i^\delta. \quad (11)$$

Cai and Sarkis [22] proposed a variant of the additive Schwarz preconditioning called the restricted additive Schwarz preconditioner. They introduced a simple and efficient change by removing the overlap in the first restriction operator as given in the following.

$$M_{RAS}^{-1} = \sum R_i^0 A_{i|L_i}^{-1} R_i^\delta. \quad (12)$$

Their main motivation was to save half the communication cost because $R_i^\delta x$ does not involve any data exchange with neighboring processors. They also observed that in comparison with the original additive Schwarz algorithm, both the iteration count and the CPU time were reduced.

VI. TEST RESULTS

In this section, we present a detailed evaluation of the proposed scheme on different test systems, ranging from 2,000 to 20,000 buses, on two multicore machines.

A. Multicore machine details

The configuration of the two machines used for scalability studies is given in Table I. Machine A has two Intel Xeon e5-2667 processors with DDR3-1600 memory and four memory channels giving a theoretical memory bandwidth of 102.2 GB/s (1600*8*4*2), while Machine B has two Intel Xeon e5-2650 v2 processors with DDR3-1866 memory and four memory channels has 119.4 GB/s theoretical memory bandwidth. To compare the scalability of the two machines, we define the following estimate for the maximum achievable scalability, $S_{\text{max}}$.

$$S_{\text{max}} = \frac{\text{Maximum achievable memory bandwidth}}{\text{Bandwidth of single processing unit}} \quad (13)$$

Note that this estimate is based on the achievable memory bandwidth as theoretical memory is never attained as the CPU is not transferring data 100% of the time. We use the STREAM benchmark [23] to calculate the maximum achievable bandwidth. This benchmark estimates the achievable memory bandwidth from four basic floating-point operations: copy, scale, add, and triad. As seen in Table I, an estimated $S_{\text{max}}$ of 3.6 and 6.4 was obtained for machine A and machine B, respectively. Thus, one can expect that a higher scalability can be achieved with machine B.

<table>
<thead>
<tr>
<th>Machine A</th>
<th>Machine B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor</td>
<td>Intel Xeon E5-2667</td>
</tr>
<tr>
<td>Processor speed</td>
<td>2.9 GHz</td>
</tr>
<tr>
<td>L3 cache/processor</td>
<td>15 MB</td>
</tr>
<tr>
<td>Physical cores/processor</td>
<td>6</td>
</tr>
<tr>
<td>Hyperthreading</td>
<td>No</td>
</tr>
<tr>
<td>Processing units</td>
<td>12</td>
</tr>
<tr>
<td>Memory size (RAM)</td>
<td>64 GB</td>
</tr>
<tr>
<td>Memory type</td>
<td>DDR3-1600</td>
</tr>
<tr>
<td>Max. memory bandwidth (theoretical)</td>
<td>102.2 GB/s</td>
</tr>
<tr>
<td>Max. achievable bandwidth</td>
<td>56.4 GB/s</td>
</tr>
<tr>
<td>Memory bandwidth (single process)</td>
<td>14.5 GB/s</td>
</tr>
<tr>
<td>Max. achievable speedup</td>
<td>3.9</td>
</tr>
</tbody>
</table>

B. Implementation details

The simulator developed in this work is capable of modeling three-phase unbalanced dynamics, unlike existing dynamics simulators that support only positive sequence representation. This three-phase representation implies a possible ninefold increase in the computational burden for the network portion. Details of the three-phase dynamics simulator can be found in [5] and [24]. The code for the developed simulator is written in C using the PETSc [25] library framework and compiled with GNU’s gcc compiler with -O3 optimization.

C. Test case details

The inventory of the test cases is given in Table II. Test case case2737sop, available with MATPOWER [26], represents the Polish 400, 220, and 110 kV networks during summer 2004 off-peak conditions. The data for case9241pegase, also available with MATPOWER, stems from the Pan European Grid Advanced Simulation and State Estimation (PEGASE) project, part of the 7th Framework Program of the European Union.
Union. Test case case22996 is a synthetic test case created by combining several MATPOWER test cases by adding tie-lines for connecting the different networks. The case labeled caseUtil is a real utility network comprising approximately 18,500 buses.

### TABLE II
**INVENTORY OF TEST CASES**

<table>
<thead>
<tr>
<th>Case Name</th>
<th>Buses</th>
<th>Generators</th>
<th>Branches</th>
</tr>
</thead>
<tbody>
<tr>
<td>case2737sop [26]</td>
<td>2737</td>
<td>399</td>
<td>3506</td>
</tr>
<tr>
<td>case9241pegase [27]</td>
<td>9241</td>
<td>1445</td>
<td>16049</td>
</tr>
<tr>
<td>case22996 [7]</td>
<td>22996</td>
<td>2416</td>
<td>27408</td>
</tr>
<tr>
<td>caseUtil</td>
<td>18500</td>
<td>2000</td>
<td>21000</td>
</tr>
</tbody>
</table>

For the dynamic simulations, all the generators were modeled by using the GENROU model [28] with an IEEE Type 1 exciter model [28], and the loads were modeled as constant impedances. The numerical integration scheme used is an implicit trapezoidal scheme with a time step of 0.008333 seconds (i.e., half-cycle for 60 Hz frequency). All the simulations on the first three test cases were run for 10 seconds with a six-cycle temporary three-phase fault applied at a bus at 1 second. For the utility network, a twelve-cycle temporary fault was applied at 0.2 seconds.

### D. Serial performance

Tables III and IV list the execution times on a single processor for a 10-second simulation using two different linear solvers. PETSc provides a direct sparse linear solver (LU factorization + triangular solves) that uses an efficient memory access scheme for storing the factored L and U matrices [29]. KLU [30] is a software package that provides a direct sparse linear solver specifically for solving linear systems that arise in circuit simulation problems and has been reported as an efficient linear solver for power system applications [31], [32]. Various reordering strategies, available with PETSc and SuiteSparse [33], were investigated to determine the optimal reordering scheme, namely, the ordering scheme resulting in the least number of nonzeros in the factored matrix. The quotient minimum degree ordering in PETSc was found to be the best reordering on the systems that we tested. In addition, a very dishonest Newton strategy is also used that updates the numerical factorization only at fault-on and fault-off times.

### TABLE III
**EXECUTION TIMES FOR SINGLE CORE USING PETSC AND KLU ON MACHINE A**

<table>
<thead>
<tr>
<th>Test System</th>
<th>PETSc</th>
<th>KLU</th>
</tr>
</thead>
<tbody>
<tr>
<td>case2737sop</td>
<td>24.4</td>
<td>27.14</td>
</tr>
<tr>
<td>9241pegase</td>
<td>93.27</td>
<td>113.27</td>
</tr>
<tr>
<td>case22996</td>
<td>144.16</td>
<td>180.39</td>
</tr>
<tr>
<td>caseUtil</td>
<td>161.86</td>
<td>203.91</td>
</tr>
</tbody>
</table>

### E. Parallel linear solver tuning

All the simulations done use a Krylov-Schwarz linear solution scheme GMRES preconditioned with a restricted overlapping additive Schwarz scheme. Scalability results with different amounts of overlap are presented in the following sections. A lagging preconditioner, or very dishonest preconditioning strategy [5], was used to update the preconditioner only at fault-on and fault-off times. A comparison with using a multifrontal direct parallel linear solver, available through the MUMPS package [34], is also presented.

The overlapping Schwarz preconditioners need the solution of the subdomain linear system

\[
A^{-1}_{i\cdot r_{i}}r
\]

during the GMRES iterations, where \( r_i = R_i^k (b - Ax) \) is the restricted error. Equation (14) can be done either by direct or by iterative linear solution schemes. Our experiments on these test cases with stationary iterative schemes such as Gauss-Jacobi, Gauss-Seidel, and successive over-relaxation (SOR) corroborate their poor convergence. In this work, we solve the linear system in Equation (14) by a direct solver (LU factorization + triangular solves). For a small number of subdomains, direct solve using LU is expensive because the subdomains are large, but the subdomain solve is robust. With more processors, the subdomain sizes become smaller, reducing the cost of direct solves. To minimize the fill-ins created by LU factorization, we use a quotient minimum degree reordering scheme.

### F. Parallel performance

Figures 3–8 show the scalability of the dynamics simulation with the Krylov-Schwarz linear solution scheme GMRES preconditioned with a restricted overlapping additive Schwarz scheme, with different amounts of overlap. For all the test cases, the Krylov-Schwarz linear solution was found to be more scalable than a parallel direct solution using MUMPS. In fact, parallel direct solution with MUMPS yielded poor scalability for all the cases tested.

Table V summarizes the peak scalability observed on the different test cases on the two machines.

### TABLE V
**PEAK SCALABILITY FOR DIFFERENT TEST CASES**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>case2737sop</td>
<td>1.65</td>
<td>4</td>
<td>3</td>
<td>3.53</td>
<td>16</td>
<td>2</td>
</tr>
<tr>
<td>9241pegase</td>
<td>2.95</td>
<td>8</td>
<td>3</td>
<td>5.06</td>
<td>16</td>
<td>3</td>
</tr>
<tr>
<td>case22996</td>
<td>3.09</td>
<td>8</td>
<td>3</td>
<td>4.21</td>
<td>16</td>
<td>3</td>
</tr>
</tbody>
</table>

More scalability was observed for larger overlaps with the overlapping Schwarz preconditioner. This behavior is typical...
with most overlapping Schwarz methods where more overlap implies a stronger preconditioner resulting in faster convergence of GMRES. However, more overlap also means an increase in the time to build the preconditioner, which may result in overall slowdown. Hence, an optimal overlap needs to be chosen that balances convergence speed and execution time. In our experiments, overlaps of 2 and 3 were found to be optimal on the different test cases. Overlaps greater than 3 did not increase the scalability because of the increased time to build the preconditioner and communication.

For case2737sop, the smallest of the test cases, as seen in Figures 3 and 4, the Schwarz preconditioners with overlap greater than 1 are more scalable compared to overlap 0 and direct parallel solver MUMPS. The max. achievable scalability on Machine A was found to be 1.6 on 4 cores, while for Machine B it was 3.53 on 16 cores. On Machine A, the scalability decreases after 4 cores for the overlapping preconditioners because the preconditioner becomes is weaker leading to more iterations for the Krylov-iterative solver resulting in more communication. For the same simulations on more than four cores on Machine B, speed up is still obtained as Machine B has more memory bandwidth.

We note that, with an overlap-0 Schwarz preconditioner, the dynamics simulation did not converge for test case 2737sop on 16 cores on Machine B. Hence, the scalability of the simulation on 16 cores with overlap 0 is missing in Fig. 4. The reason for this divergence is that the overlap-0 based preconditioner is weak on 16 cores for this test case. This is a known issue with Schwarz-based methods on larger number of cores, i.e., Schwarz-based preconditioner with smaller overlap become weaker as the number of cores increase.

The max. achievable scalability for case9241pegase was found to be 2.95 on 8 cores and 5.06 on 16 cores with overlap-3 for Machine A and Machine B, respectively. As
seen in Figures 5 and 6, the Schwarz preconditioners with overlap greater than 1 are more scalable compared to overlap 0 and direct parallel solver MUMPS. case9241pegase being considerably larger, almost three-folds, than case2737sop, the decrease in the speed-up was for Schwarz-preconditioners with overlaps greater than 1 was observed beyond 8 cores.

Similar scalability behavior as case9241pegase was seen for case22996 as shown in Figures 7 and 8. On Machine A, a peak scalability of 3.09 was observed on 8 cores, while for Machine B it was 4.21 on 16 cores.

G. Real utility test case

Figure 9 shows the scalability of dynamics simulations for different linear solvers on the utility network on Machine A. A maximum speedup of 2.13 was obtained on 8 cores with the Krylov-overlapping Schwarz method with an overlap of 3. For dynamics simulations on 8 cores and beyond, the solution did not converge for overlap-0 and hence the scalability results are not shown. Our conjecture for this behavior is that the preconditioner is really weak leading to the divergence of the Krylov linear solver. This suggests that using an overlap greater than 1 results in a more robust preconditioner.

VII. CONCLUSIONS

This paper presented a Krylov-Schwarz linear solution scheme for accelerating parallel dynamics simulation. The proposed approach shows good scalability on different power system test cases ranging from 2,000 to 20,000 buses, including a real utility network. Performance of the linear solution scheme shows that an overlap of 2 or 3 is optimal for overlapping Schwarz preconditioning for the given test cases. A metric based on memory bandwidth was presented for estimating the maximum achievable speedup of a computing architecture to guide scalability testing of parallel dynamics simulations.

ACKNOWLEDGMENTS

This work was supported by U.S. Dept. of Energy, Office of Electricity, Advanced Grid Modeling Research, under Contract DE-OE0000624.

REFERENCES


[27] Shrirang Abhyankar received his M.S degree (2006) and his Ph.D degree (2011) in electrical engineering from Illinois Institute of Technology, Chicago. He is currently a computational engineer in the Energy Sciences Division at Argonne National Laboratory. His research interests include scalable algorithms for large-scale transient stability analysis, combined electromechanical and electromagnetic simulation, and co-simulation of transmission-distribution dynamics. He is a co-developer of the PETSc library.


APPENDIX A
PETSC: PORTABLE EXTENSIBLE
TOOLKIT FOR SCIENTIFIC COMPUTATION [25]
The PETSc package consists of a set of libraries for creating parallel vectors, matrices, and distributed arrays, and scalable linear, nonlinear, and time-stepping solvers. In this work, we used the linear solver GMRES and the overlapping additive Schwarz method (ASM) preconditioner class (ASM) available in PETSc. The ASM preconditioner class includes several variants with the default being the restricted additive Schwarz preconditioner. In addition, we used a fixed-step implicit trapezoidal integration scheme available through PETSc’s time-stepping library TS. PETSc also has a plug-in architecture for third-party solvers and has interfaces available for several direct linear solvers, including SuperLU Dist [35], MUMPS [34], and KLU [30].

Shrirang Abhyankar received his M.S degree (2006) and his Ph.D degree (2011) in electrical engineering from Illinois Institute of Technology, Chicago. He is currently a computational engineer in the Energy Sciences Division at Argonne National Laboratory. His research interests include scalable algorithms for large-scale transient stability analysis, combined electromechanical and electromagnetic simulation, and co-simulation of transmission-distribution dynamics. He is a co-developer of the PETSc library.
Emil M. Constantinescu received his Ph.D. degree in computer science from Virginia Tech, Blacksburg, in 2008. He is currently a computational mathematician in the Mathematics and Computer Science Division at Argonne National Laboratory. His research interests include numerical analysis of time-stepping algorithms and their applications to energy systems.

Barry F. Smith received his B.S. in mathematics from Yale University in 1986 and his Ph. D. in mathematics from the Courant Institute/New York University in 1990. He is a senior computational mathematician at Argonne National Laboratory and leads the PETSc software development team.

Alexander J. Flueck received his B.S. degree (1991), M.Eng. degree (1992) and Ph.D. degree (1996) in electrical engineering from Cornell University. He is currently an associate professor at the Illinois Institute of Technology in Chicago. His research interests include three-phase transient stability, electromagnetic transient simulation, autonomous agent applications to power systems, transfer capability of large-scale electric power systems, contingency screening with respect to voltage collapse, and parallel simulation of power systems via message-passing techniques on distributed-memory computer clusters.

Daniel A. Maldonado received his B.S. degree from the Universitat Politecnica de Valencia, Valencia, Spain, and M.Eng degree from the Illinois Institute of Technology, Chicago. Currently, he is a Ph.D. candidate at the Illinois Institute of Technology. His research interests include transient stability simulation, state estimation, and optimal control.