

Real-Time Power System Dynamics Simulation using a Parallel Block-Jacobi Preconditioned Newton-GMRES scheme

Shrirang Abhyankar

Argonne National Laboratory

Alexander Flueck

Illinois Institute of Technology

November 11, 2012

Real-time Dynamics Simulation

Motivation and challenges

- Motivation: Move from “static” to “dynamic” mode of operation
 - Faster assessment of dynamic phenomenon such as cascading outages
 - Support for PMU data
 - Dynamic control of grid
- Challenge: Dynamics simulation of large systems is too slow for an online application



Power Grid Dynamics Formulation and Solution

Nonlinear Differential-Algebraic
Power Grid Model

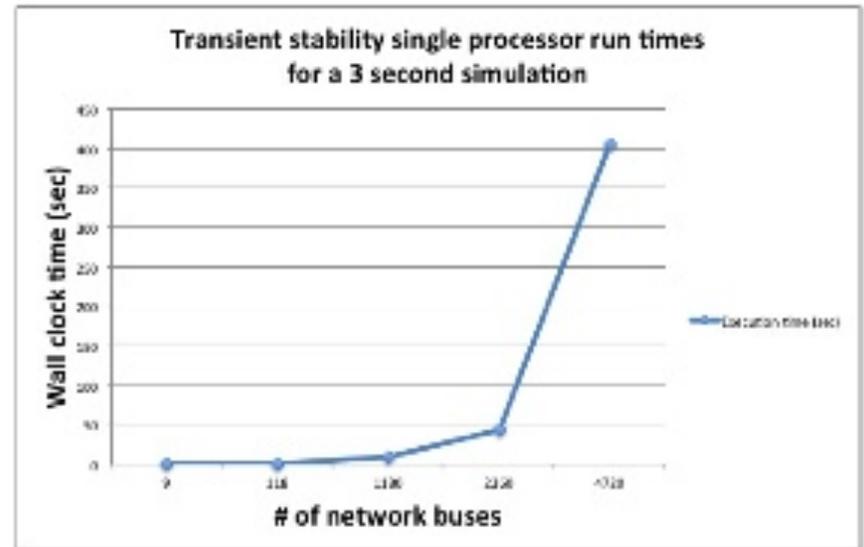
$$\frac{dx}{dt} = f(x, y) \quad \text{Machines}$$

$$0 = g(x, y) \quad \text{Network}$$

Numerical requirements

- Numerical Integration
- Nonlinear solver
- **Linear solver**

Execution times on a **single core**
for different sized systems for a
temporary fault scenario



- Exponential increase in execution time as system size increases.
- Solution in real-time can be unachievable for large systems such as utility networks or regional interconnections.

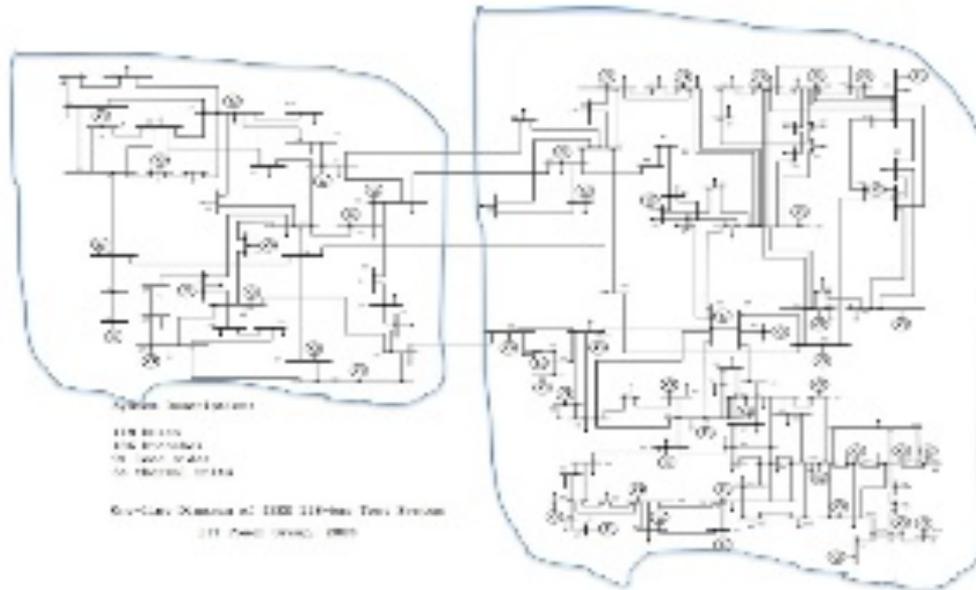
Parallel dynamics simulation: Overview

- Most of the research done in 1990s
- Parallel-in-space (Spatial Decomposition)
 - A. Bose, L. Decker
- Parallel-in-time (Temporal decomposition)
 - F. Alvarado
- Waveform relaxation (Spatial and temporal decomposition)
 - M. Ilic, A. Bose, M. Crow
- Survey paper
 - High performance computing in power systems (D. Falcao)



Parallel-in-space or spatial decomposition approach

- Divide and conquer spatially
- How to get a decent decomposition? (Graph partitioning problem)
 - We use graph partitioning package ParMetis (developed at U. Minnesota by G. Karypis)
 - Build the adjacency graph for the network structure (generators, loads naturally decoupled).
 - Additional vertex weights for nodes with generators.



Example spatial decomposition of 118 bus system into two subdomains

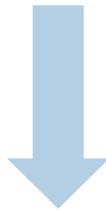


Parallel-in-space dynamics equations

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

DAE equations for each subdomain

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



Implicit trapezoidal discretization

$$x_p^{n+1} - x_p^n - \frac{\Delta t}{2} (f(x_p^{n+1}, y_p^{n+1}) + f(x_p^n, y_p^n)) = 0$$

Nonlinear equations for each subdomain to be solved by Newton's method at each time step

$$0 = g(x_p^{n+1}, y_p^{n+1}, y_c^{n+1})$$

$$J(X^i) \Delta X^{i+1} = -F^i$$

Linear system to be solved at each Newton iteration is the main computational bottleneck

Solving $Ax = b$ via Krylov subspace methods

- Work on a basis of successive matrix powers times the initial residual.

$$K(A, r_0) = \text{span}\{r_0, Ar_0, A^2r_0, \dots, A^{n-1}r_0\}$$

- Iterative solution through approximations formed by reducing residual over the Krylov subspace.
- Several established algorithms: CG, GMRES, BiCG, Lanczos, BCGS
- *“Iterative Methods for Sparse Linear Systems”* – Y. Saad
- Efficient in parallel
 1. No factorization involved
 2. Consist of only matrix-vector products and reductions.
- Convergence depends on spectral properties of operator matrix A
 - Need preconditioners to accelerate convergence for most problems.



Generalized Minimum Residual Method (GMRES)

- Krylov subspace method for solving $Ax=b$ where A is square, nonsingular, and nonsymmetric.
- Approximates solution by minimizing the residual over the Krylov subspace

GMRES algorithm

Start/Restart: Compute $r_0 = b - Ax_0$ and $v_1 = r_0 / (\beta := \|r_0\|_2)$

Arnoldi Process: generate V_m (orthonormal basis) and \bar{H}_m (Hessenberg matrix)

Minimization: Compute $y_m = \operatorname{argmin} \| \beta e_1 - \bar{H}_m y \|_2$

Update: $x_m = x_0 + V_m y_m$

Convergence check: If $\|r_m\|_2 \leq \varepsilon \|r_0\|_2$ stop else set $x_0 := x_m$



Preconditioners and Parallel Block-Jacobi Preconditioner

- Convergence of GMRES depends on the eigen spectrum of the linear operator.
- A preconditioner transforms the linear system into another system with better spectral properties.

- Left preconditioning $M^{-1}Ax = M^{-1}b$
- Right Preconditioning $AM^{-1}x = b$
- Split Preconditioning $M_1^{-1}AM_2^{-1}x = M^{-1}b$

- **Parallel Block-Jacobi Preconditioning**

- Formed by using the “diagonal block” on each subdomain.

$$0 \begin{bmatrix} J_1 & J_2 \\ J_3 & J_4 \end{bmatrix}$$

Jacobian on 2 cores

$$0 \begin{bmatrix} J_1^{-1} & \\ & J_4^{-1} \end{bmatrix}$$

Block-Jacobi preconditioner

- No communication for building or applying the preconditioner
- Can choose the factorization, reordering independently on each block.
- We use LU with Quotient Minimum Degree ordering on each block.



Test case and hardware description

- 2383 bus system (from MatPower)
 - 327 generators, 2896 branches, 25281 MW.
 - All generators modeled by GENROU (6th order) model
 - All generators have IEEE Type 1 exciter.
 - Loads modeled as constant impedance
- Hardware and software
 - Shared memory machine with four 2.2 GHz AMD Opteron 6274 processors.
 - 16 cores/processor to give a total of 64 cores.
 - Code developed using PETSc and compiled with GNU compiler with `-O3` optimization.
- Test scenarios
 - Three-phase fault on bus 185 for 0.1 seconds (stable case)
 - Three-phase fault on bus 18 for 0.1 seconds (unstable case)
- Performance of Block-Jacobi Preconditioned Newton-GMRES compared with parallel direct solver MUMPS.



Dynamics simulator details

- Full three-phase representation of the network
 - Unsymmetrical disturbances.
 - Large single phase loads.
 - Couple with distribution system.
- Uses a parallel-in-space approach

* http://www.mcs.anl.gov/~abhyshr/downloads/thesis/shri_phd_thesis.pdf



Portable Extensible Toolkit for Scientific Computation (PETSc) www.mcs.anl.gov/petsc

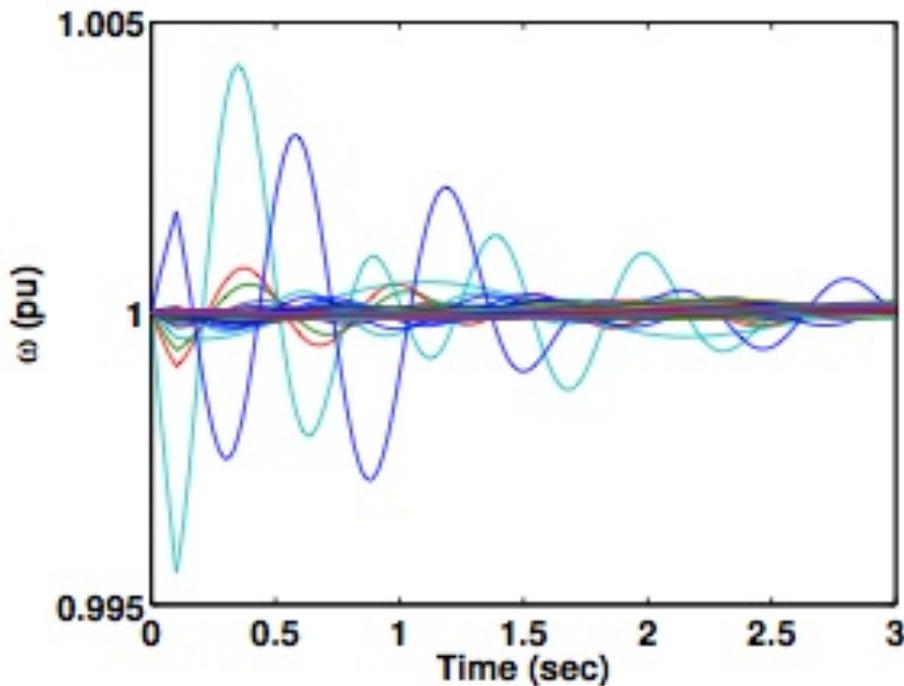
- “Open-source” numerical library for large-scale parallel computation.
- Top 100 R&D award in 2009.
- Portability
 - Distributed memory, shared memory, GPUs.
 - Unix, Linux, MacOS, Windows
 - 32/64 bit, real/complex, single/double/quad precision.
 - C, C++, Fortran, Python, MATLAB.
- Extensibility
 - ParMetis, SuperLU, MUMPS, HYPRE, UMFPACK, Sundials, PTScotch (over 20 packages)
 - Child packages – TAO, SLEPc, libMesh
- Toolkit for Scientific Computation
 - Lots of parallel linear solvers and preconditioners.
 - Parallel nonlinear solvers.
 - Parallel timestepping (DAE and ODE) solvers.



Simulation results for stable case

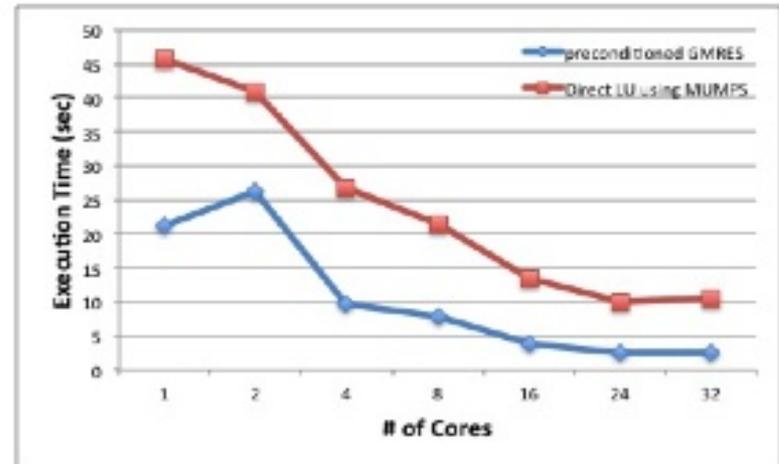
3-phase fault on bus 185 for 0.1 seconds

Generator speeds

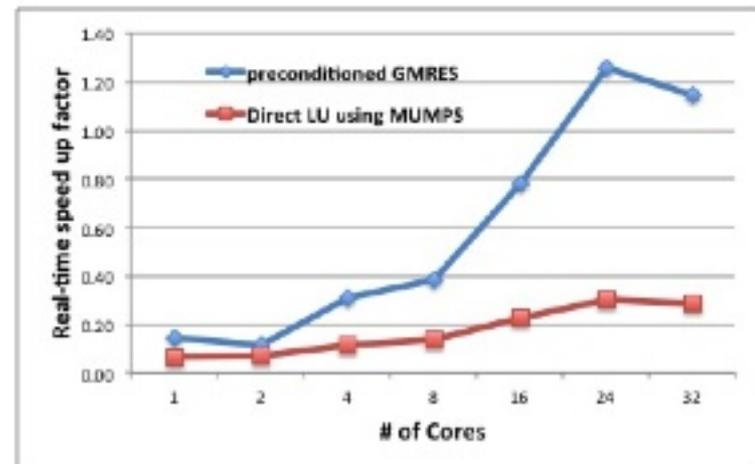


$$S_r = \frac{\text{Simulation Time}}{\text{Execution Time}}$$

Comparison of Execution times



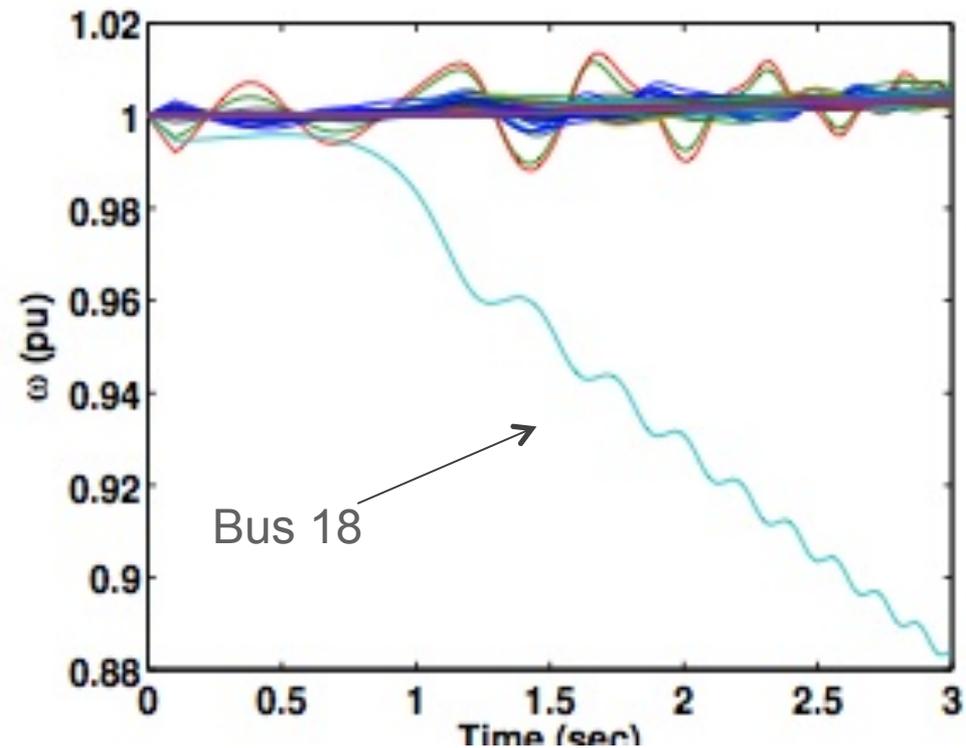
Comparison of real-time speedup factor



Simulation results for unstable case

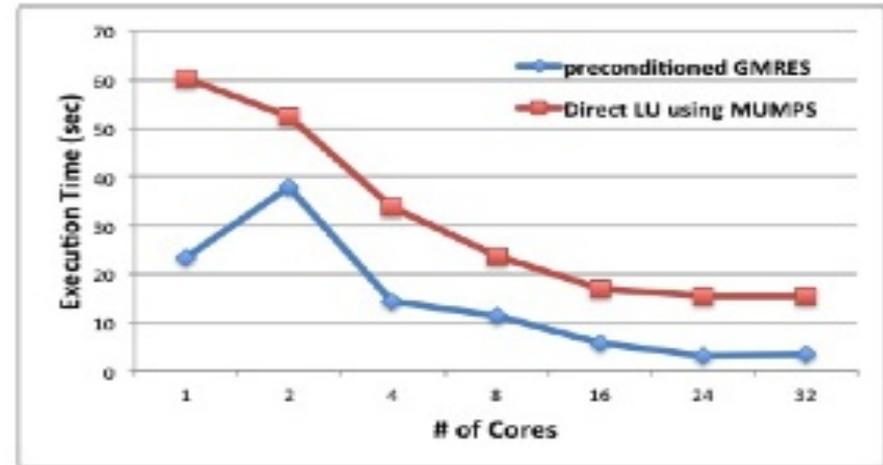
3-phase fault on bus 18 for 0.1 seconds

Generator speeds

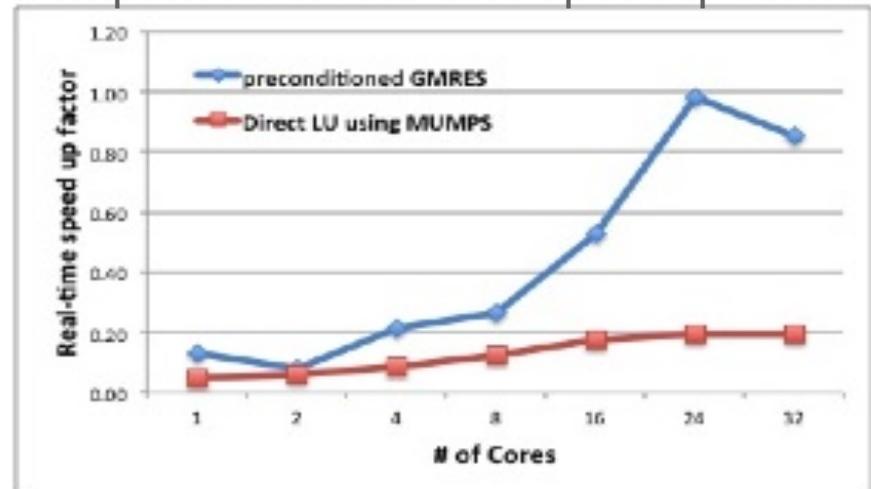


$$S_r = \frac{\text{Simulation Time}}{\text{Execution Time}}$$

Comparison of Execution times



Comparison of real-time speedup factor



Summarizing

- Presented a preconditioned GMRES scheme that is able to realize real-time speed for the test case presented.
- Moving from “static” to “dynamic” operation mode would require a thorough analysis of different algorithms under different operating conditions.
- HPC libraries like PETSc can aid in the rapid development and testing of different algorithms for power grid applications



Questions?

