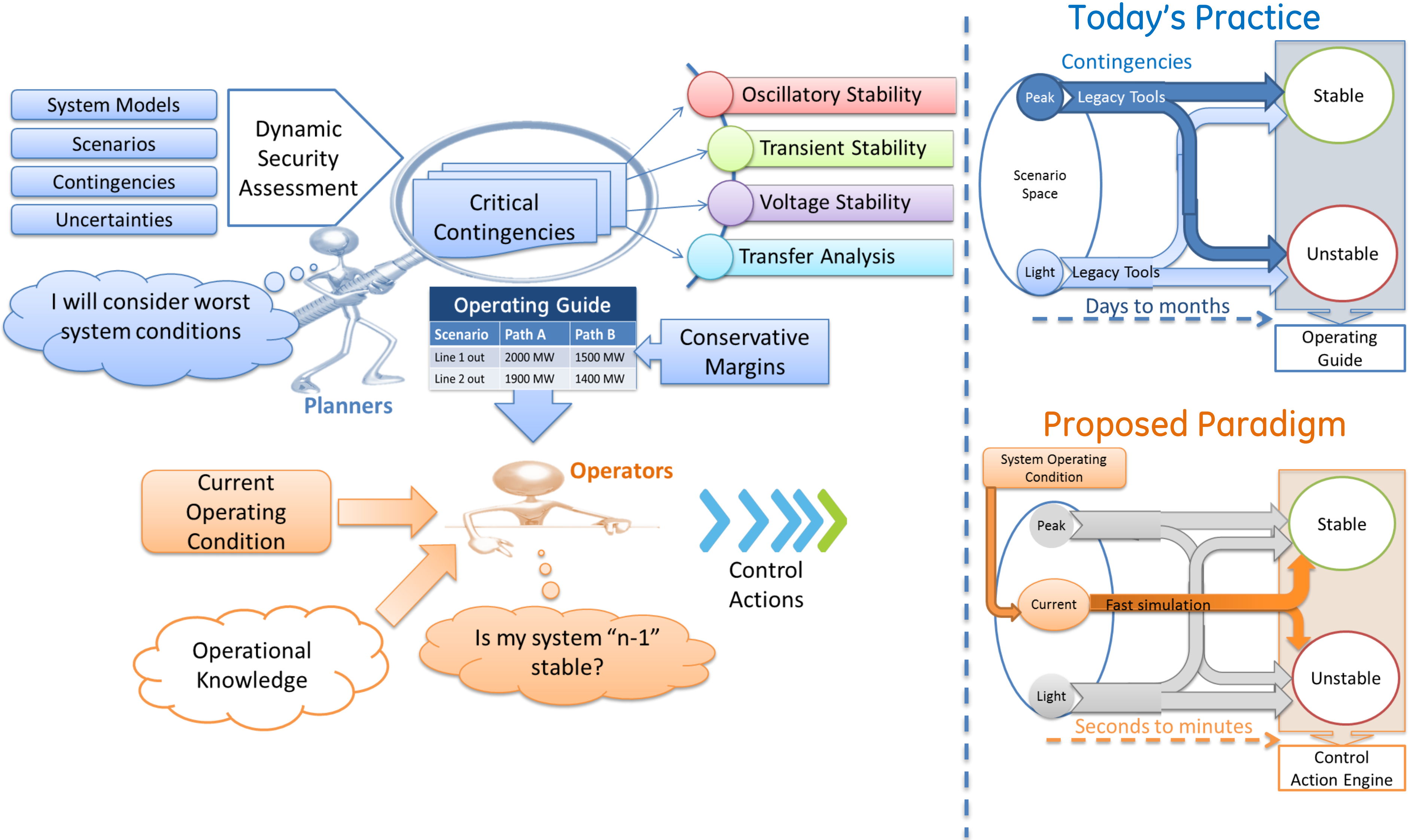
Fast Dynamic Simulation-Based Small Signal Stability Assessment & Control

Dr. Naresh Acharya
GE Global Research
[Collaborating Organizations: GE Energy Consulting, PNNL, SCE]
June 18 2014

Context/Vision



Scenario	Path A	Path B
Line 1 out	2000 MW	1500 MW
Line 2 out	1900 MW	1400 MW

Presentation Outline

- Project Objectives
- Technical Approach
  - Parallelization of PSLF
  - Alternative DAE Modeling
  - Fast Contingency Screening
  - Oscillation Damping Control
- Technical Accomplishments
- Project Team

Project Objectives

- **Apply advances in high performance computing techniques to develop fast Contingency Screening and Control Action Engine (FSCAE) for proactive small signal stability assessment, prediction and control.**
  - Develop mathematical and high-performance computing (HPC) techniques applicable to power system fast dynamic simulation.
  - Implement HPC techniques in power system dynamic simulation software
  - Develop fast contingency screening method
  - Develop oscillation damping control method
  - Verify and validate speed enhancement of dynamic simulation and decision making methods.

Technical Approach

Accelerated Assessment

Today's practice

Contingencies	C1	C2	C3	C4
Processors	P1	P2	P3	P4
Time (mins.)	10	9	8	9

Minimum time to make decision -> 10 minutes

Proposed approach

Advanced numerical solvers

Contingencies	C1	C2	C3	C4
Processors	P1	P2	P3	P4
Time (mins.)	3	2	2	2

Minimum time to make decision -> 3 minutes

Predictive Capability

Contingency Screening

- Reduce contingency space
- Estimation in frequency domain & validation in time domain using HPC

Proactive Decision

- Proactive guidance to operators to ensure system stability
- Control based on operating point adjustment

Parallelization Approaches - Pros & Cons

Approach 1: Use parallelization under current PSLF architecture

- **Pros**
  - Faster to implement
  - Less code changes
- **Cons**
  - Speed gains are limited by the speed of slowest loop on current program architecture
  - Larger changes can be challenging and require significant modifications

Approach 2: Alternative DAE modeling (implicit integration)

- Pros
  - Expect greater speed gains than approach 1
  - Can be used as the basis for the development of other tools (small signal analysis)
- Cons
  - Slower to implement
  - More code changes and more code development
  - Need to reformulate solution approach in PSLF

Parallelization of PSLF

Challenges faced

- Program is already well written and optimized
- Complex program structure and legacy code (program has been written over 30 years)
- Overhead costs of parallelization methods will adversely impact performance on small cases
- Replacement of linear solver involves significant changes in the core

Selecting an appropriate solver for the problem at hand

- Literature review (and preliminary results) have indicated that for current power system power system matrix sizes
  - Direct methods have superior performance over iterative methods
  - Serial solvers are faster than parallel ones
- As problems grow larger, iterative methods are expected to outperform direct methods

PSLF core architecture improvements

- The most effective way to reduce the solution speed of PSLF dynamics is a combination of two techniques
  - Parallelization of ODEs
  - Fast linear solver (Network)
- Solver speed will be directly dependent on matrix sparsity structure and problem size

Preliminary results

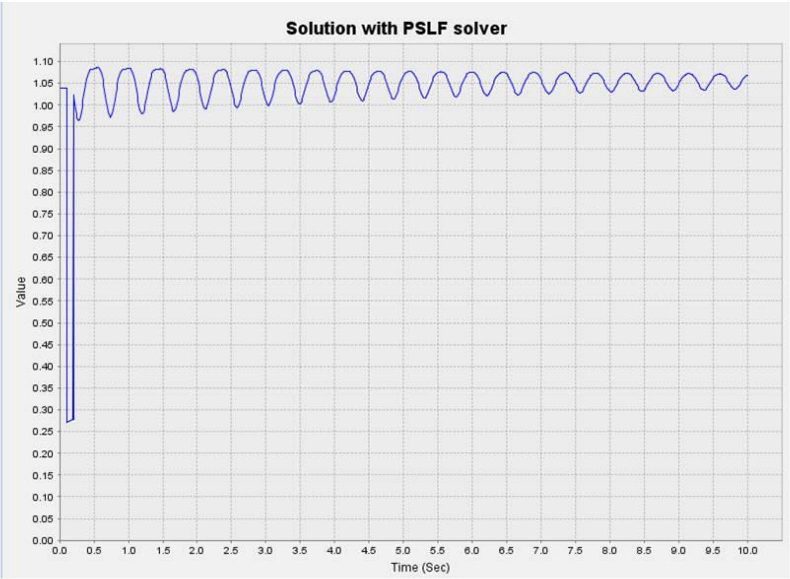
- Execution time of a 1s simulation on an real size system

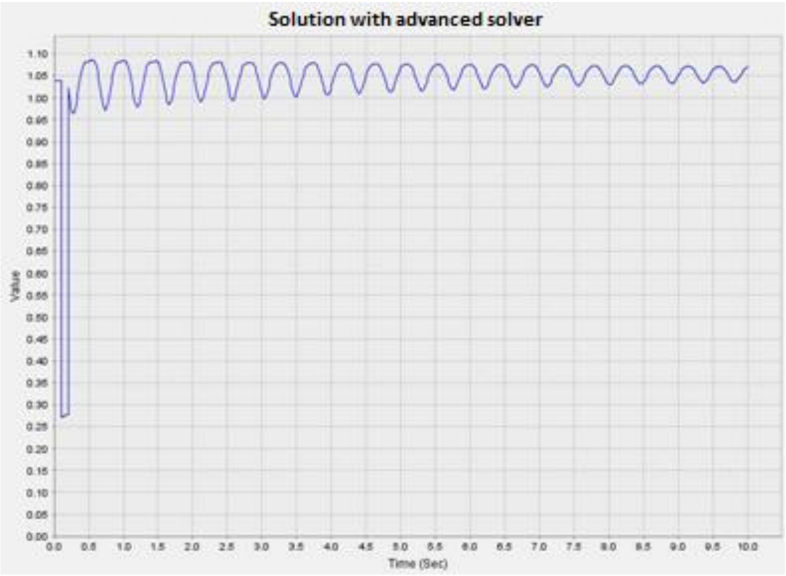
Run	Serial	Parallel (2 threads)	Parallel (3 threads)
1	11.10s	9.88s	10.22s
2	11.91s	9.71s	10.15s
3	11.60s	9.78s	10.21s
Average	11.53s	9.79s	10.19s
Gain	-	Reduction of ~15.1%	Reduction of ~12.7%

Conclusion: A more substantial performance gain will require additional modifications in the PSLF solution scheme

Linear solver replacement

- Successful code replacement (large modifications in the program)
- Solution accuracy confirmed on small case





Achievements

- Current parallel implementation leverages existing architecture of PSLF
  - Faster to incorporate
  - Utilizes extensive model database
  - Reduces the chances of errors
- Identification of faster solvers that could improve factorization speed significantly (nearly 30x)
- Limitations on speed gains are mainly due to
  - Speed of serial loops conflicting with overhead costs in parallelization
  - System factorization may not occur many times, thus improvements in factorization may not be very noticeable

Lessons Learned

- Speed improvements in the PSLF dynamic simulation
- Code modularity facilitates future solver replacements (very important)
  - Additional functionality as a byproduct of effort
- Understanding of current state of the art solvers

For more substantial speed gains, a change is solution architecture is required (integration methods/DAE)

Parallelization Approaches - Pros & Cons

Approach 1: Use parallelization under current PSLF architecture

- Pros
  - Faster to implement
  - Less code changes
- Cons
  - Speed gains are limited by the speed of slowest loop on current program architecture
  - Larger changes can be challenging and require significant modifications

Approach 2: Alternative DAE modeling (implicit integration)

- Pros
  - Expect greater speed gains than approach 1
  - Can be used as the basis for the development of other tools (small signal analysis)
- Cons
  - Slower to implement
  - More code changes and more code development
  - Need to reformulate solution approach in PSLF

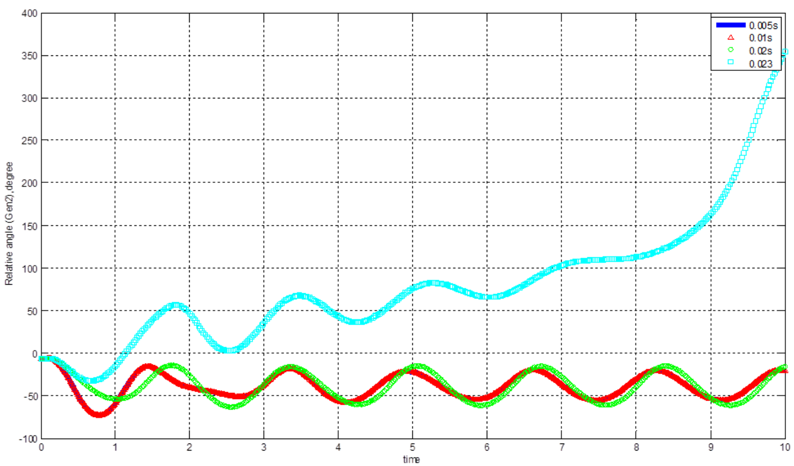
Alternative DAE Modeling

Motivations

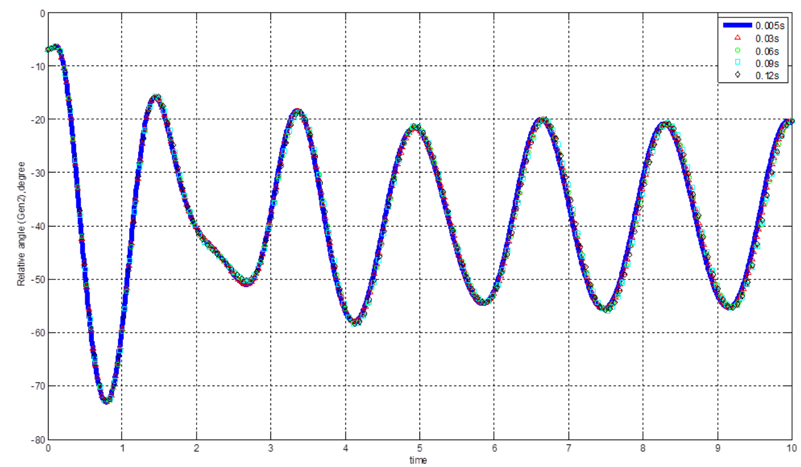
- Computationally intensive to perform dynamic simulations
  - Most commercial tools use explicit integration method, calculating differential and algebraic equations alternatively
  - A small time step is required to ensure numerical stability
  - A few times slower than real time
- Research Objective
  - Faster-than-real-time dynamic simulation powered by HPC techniques
    - Implicit integration using Trapezoidal rule
    - A time stacking method
    - Faster linear solvers
    - Adaptive time stepping with much larger time steps

Advantages of Implicit Integration

- Has better numerical stability
- Enables larger time steps for simulation



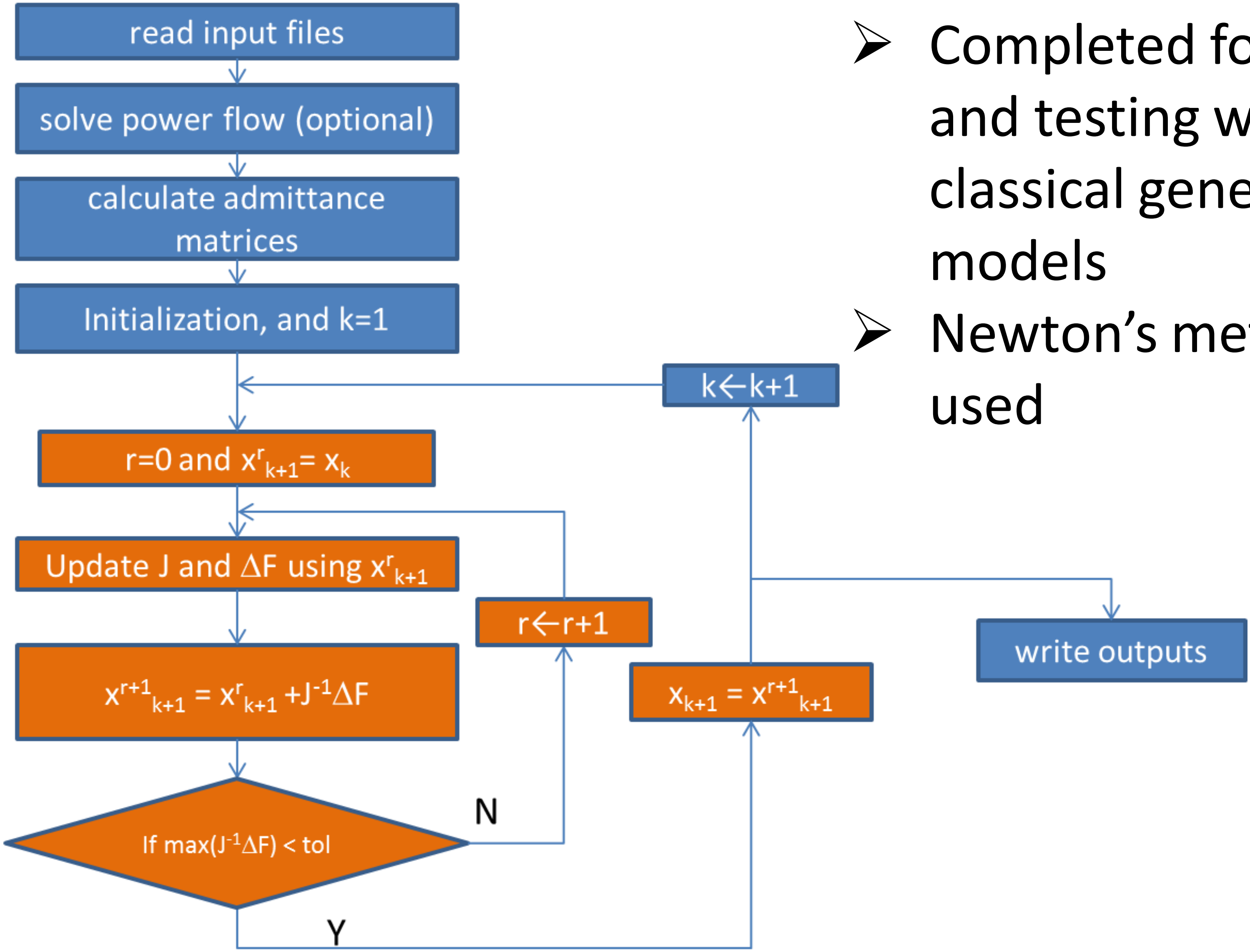
Modified Euler Method
Max allowable time step:
0.01 s



Trapezoidal method
Max allowable time step:
0.134 s

Tested on a 2-area system

Flowchart for Implicit Integration (Single Time Step)



- Completed formulation and testing with classical generator models
- Newton's method is used

Proposed Approach

- Development of a time-stacking method for solving multiple steps simultaneously
  - Combine discretized differential equation and algebraic equation
  - "Stack" multiple time steps for simultaneous solution

Sequential time-stepping process:

$$\begin{cases} x_{k+1} = f(x_k, x_{k+1}, y_k, y_{k+1}) \\ 0 = g(x_{k+1}, y_{k+1}) \end{cases}$$

↓

$$\begin{cases} x_{k+2} = f(x_{k+1}, x_{k+2}, y_{k+1}, y_{k+2}) \\ 0 = g(x_{k+2}, y_{k+2}) \end{cases}$$

↓

...

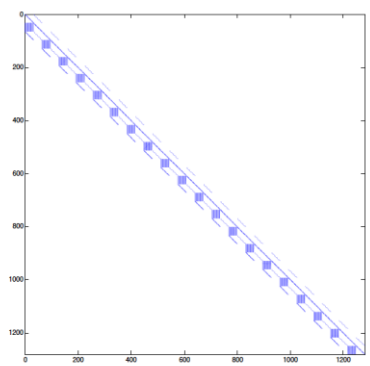
$$\begin{cases} x_{k+m} = f(x_{k+m-1}, x_{k+m}, y_{k+m-1}, y_{k+m}) \\ 0 = g(x_{k+m}, y_{k+m}) \end{cases}$$

Time-stacking method:

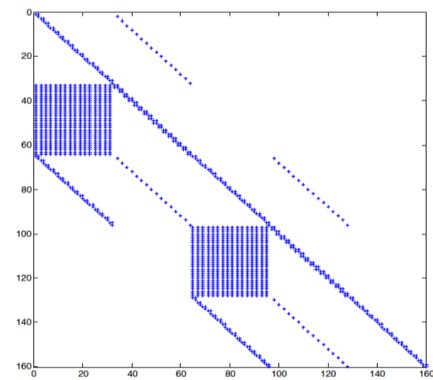
$$\begin{cases} x_{k+1} = f(x_k, x_{k+1}, y_k, y_{k+1}) \\ 0 = g(x_{k+1}, y_{k+1}) \\ x_{k+2} = f(x_{k+1}, x_{k+2}, y_{k+1}, y_{k+2}) \\ 0 = g(x_{k+2}, y_{k+2}) \\ \dots \\ x_{k+m} = f(x_{k+m-1}, x_{k+m}, y_{k+m-1}, y_{k+m}) \\ 0 = g(x_{k+m}, y_{k+m}) \end{cases}$$

Identification of Better Linear Solvers for the Time-stacking Method

- An example of Jacobian matrix derived from the time-stacking method for a 16g68b system
  - Matrix properties: real, sparse, non-symmetric, non-diagonally dominant
  - With a large condition number: 8.668x10^6



Size: 1280x1280, nnz=14016

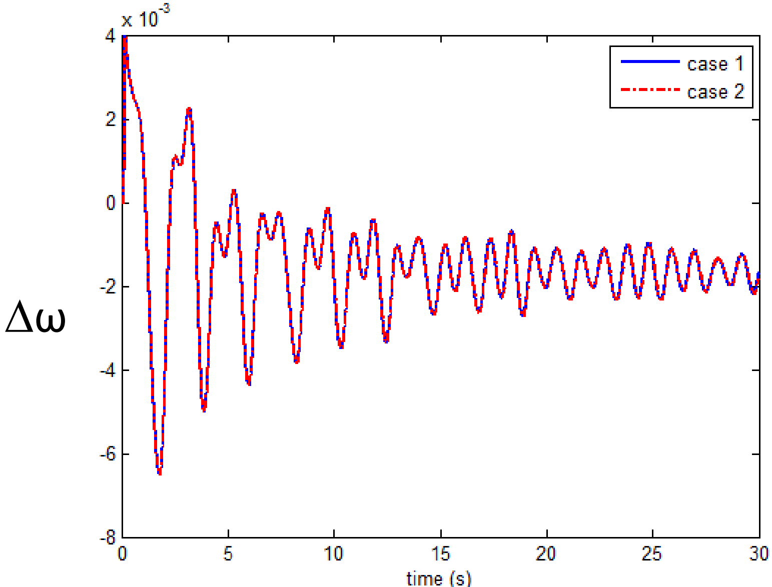


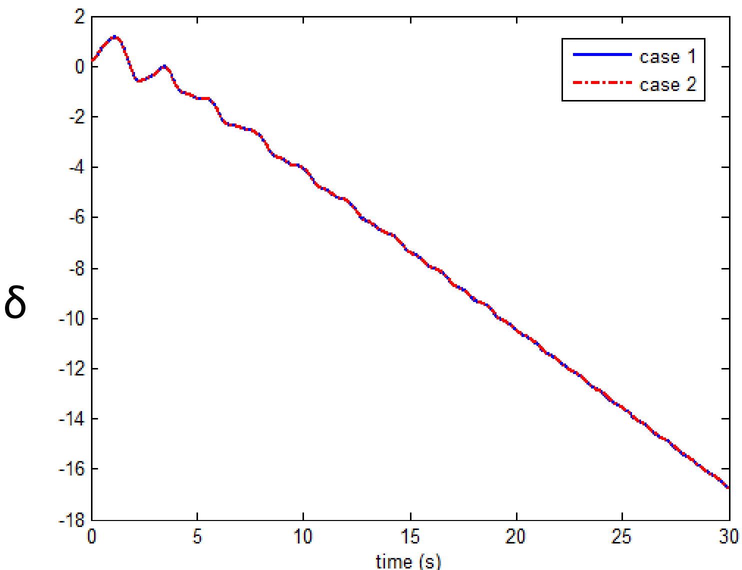
Zoom-in view

- Direct solver vs. iterative solvers (averaged 10,000 runs)
  - Sparse LU (UMFPACK): 0.0165 sec
  - BiCGSTAB + ILU preconditioner: 0.0190 sec (1 iteration, tol=1e-8)
  - GMRes + ILU preconditioner: 0.0266 sec (2 iterations, tol=1e-8)
- It is expected iterative solvers outperform direct solvers for a much larger Jacobian matrix, using multiple processors

Adaptive Time Stepping Method

- The time step is adjusted based on
  - local error estimate $k\left|y_{n+1}^{P}-y_{n+1}\right|$
  - performance of the Newton corrector iteration
  - switching events and faults
- Comprehensive logics used to adaptively change the time stepping
  - 10%~30% speedup observed from various testing





Case 1: constant time step, 0.01s

Case 2: adaptive time step, up to 0.1s

Accomplishments & Ongoing Tasks

- Accomplishments
  - Proved the concept of the time-stacking method using classical generator model
  - Compared the computational complexity of using reduced and full admittance matrices
    - 3 times speedup observed when using full admittance matrix with implicit integration
  - MATLAB code developed
  - Tested and compared direct and iterative solvers
    - Direct solver works best for solving I=Y*V, for the traditional explicit integration method, e.g., PSLF
    - Iterative solvers perform equally well for the time stacking method

- Ongoing work
  - Adding more detailed generator model and controllers to the software code
    - GENTPJ, EXAC2, IEEEG1
    - Jacobian matrices derived
  - Developing parallel version of FORTRAN code for testing the computation speed
  - Investigating techniques to improve convergence
    - Better initial values
    - Dishonest or very dishonest Newton's method

Fast Contingency Screening for Small Signal Stability

Small-Signal Security Assessment

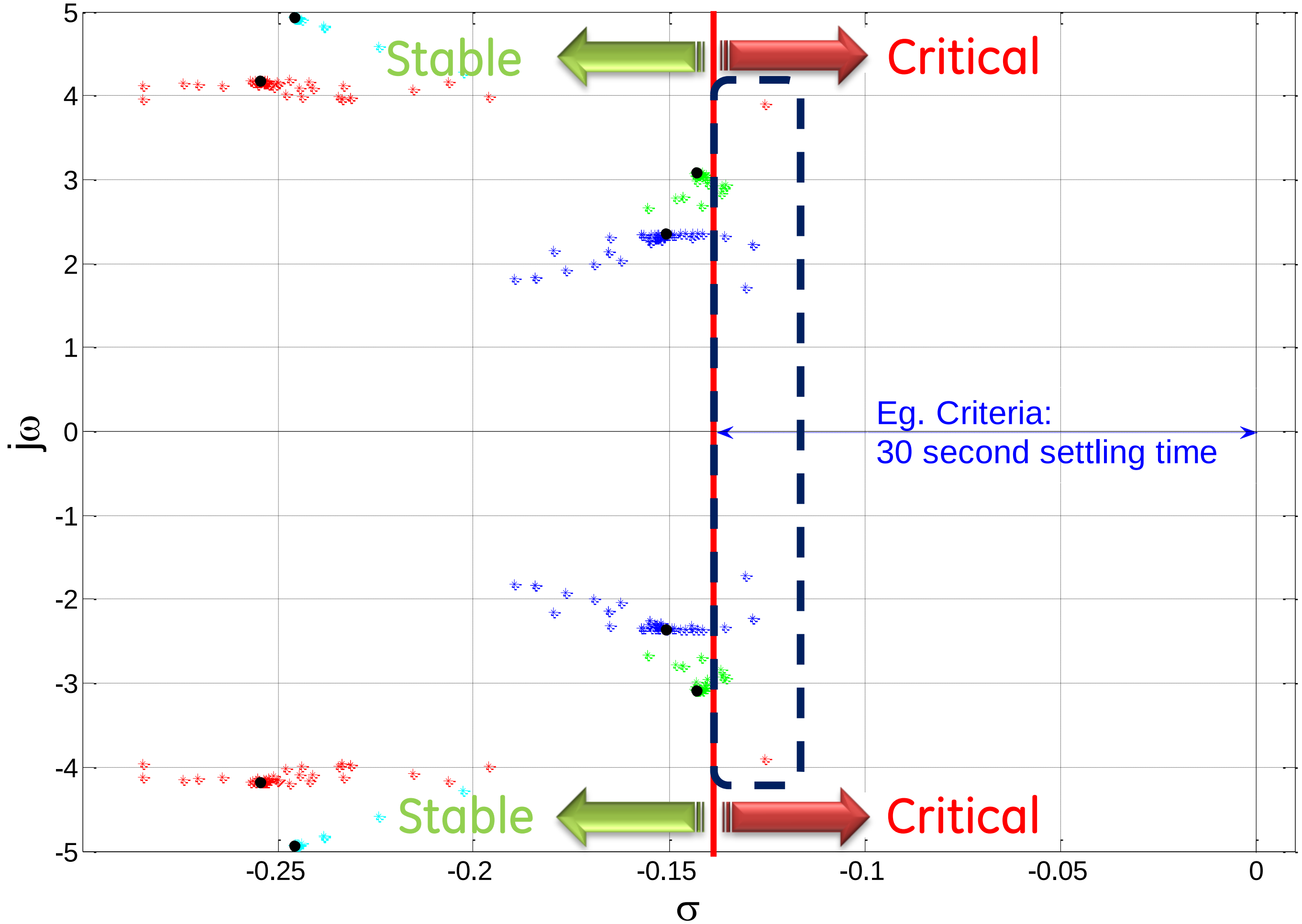
Today's Practice

- Small-signal stability analysis under set of contingencies for range of operating conditions
- System is small-signal secure if damping/settling time of all critical oscillatory modes is within a required threshold
- Offline study: during planning considering worst case system conditions (e.g. summer peak)
- Brute force: Time domain or eigenvlaue analysis for all possible contingencies and conditions

Challenges & Need for Real-Time

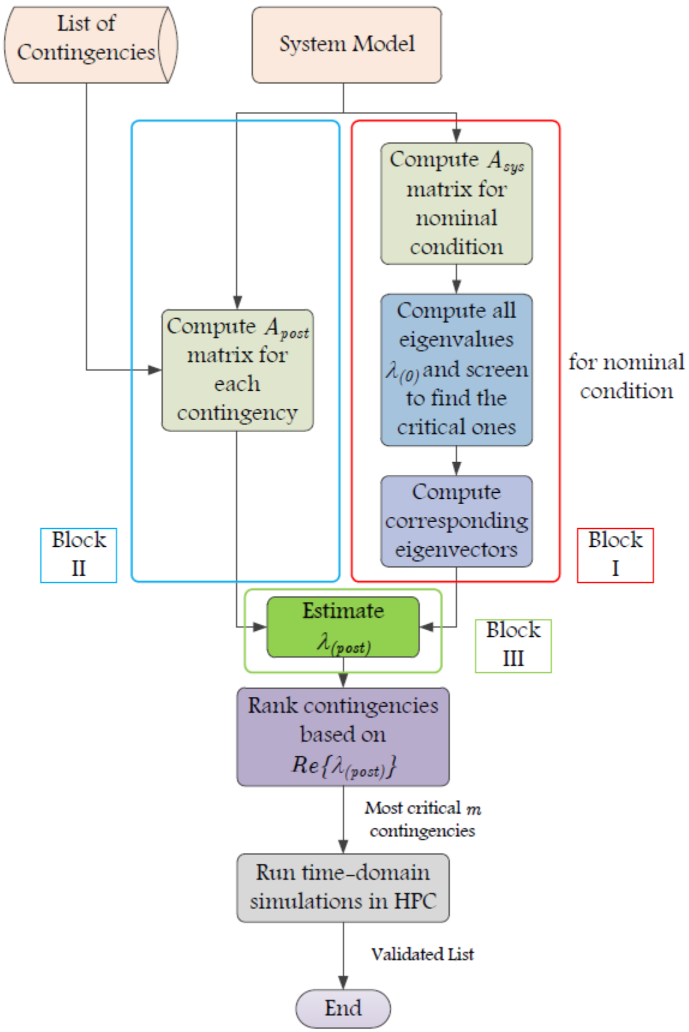
- Large number of contingencies
- Long simulation time
- Evaluating large number of contingencies using time-domain simulations or eigenvalue computation is extremely time consuming & infeasible for large systems in real-time
- Dimensionality; matrix inversion;
- Assessment is not enough – for violating contingencies control solution is needed

Contingency Screening (16-machine e.g.)



Screening will reduce the contingency space to be analyzed in real-time

Fast Contingency Screening & Ranking



Block I – Compute linearized system matrix, eigenvalues & eigenvectors for base case

A_{sys} , $\lambda_{i(0)}$, ψ_i , ϕ_i

Block II – Compute linearized system matrix (Apost) for each contingency

A_{post}

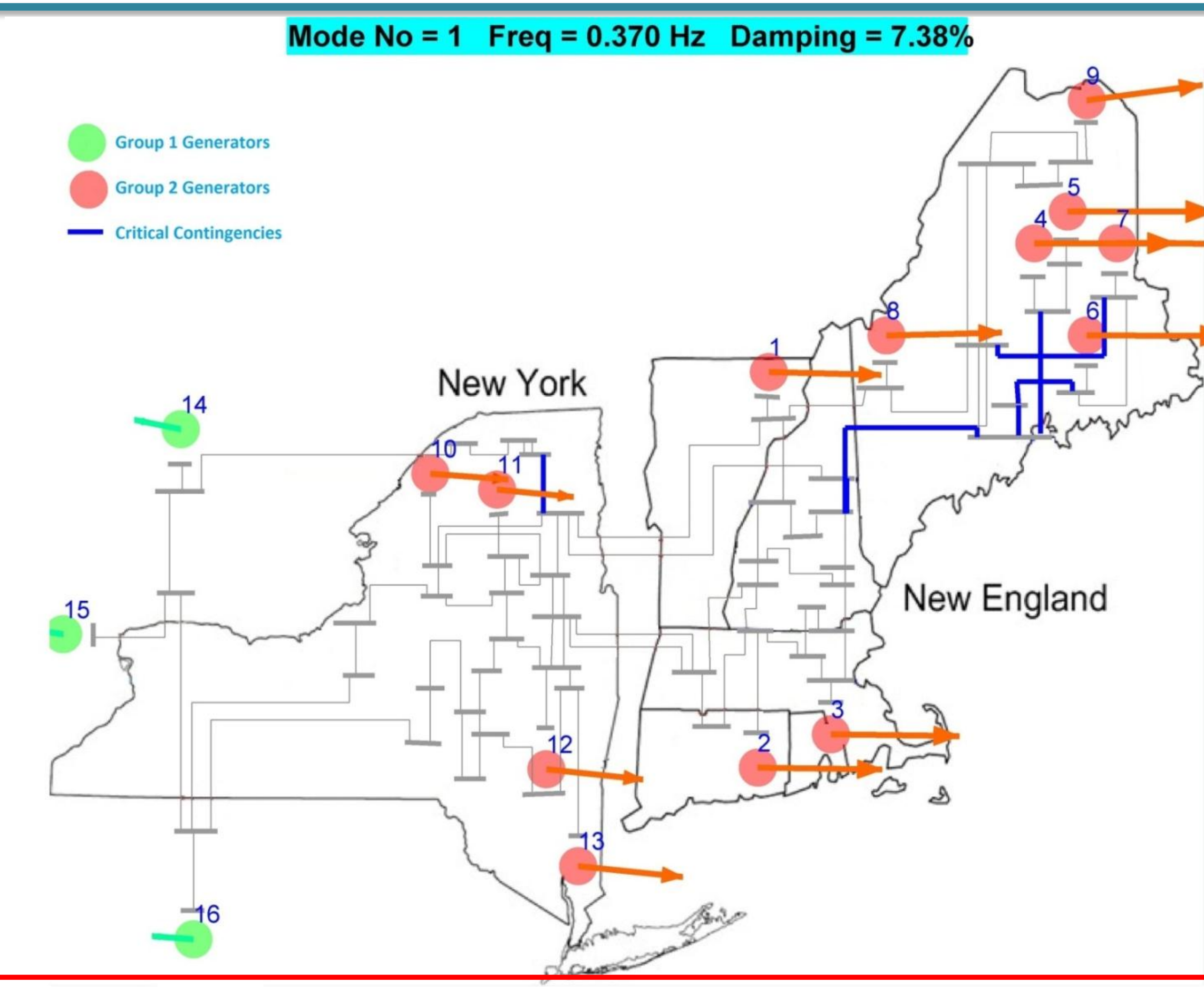
Block III – Estimate post contingency eigenvlaues based on 1st and 2nd order

$$\lambda_{i(post)} \cong \lambda_{i(0)} + \left.\frac{\partial \lambda_i}{\partial \gamma}\right|_0 \Delta\gamma + \frac{1}{2!}\left.\frac{\partial^2 \lambda_i}{\partial \gamma^2}\right|_0 (\Delta\gamma)^2$$

$$\frac{\partial \lambda_i}{\partial \gamma}\Delta\gamma \cong \frac{\psi_i \Delta A \phi_i}{\psi_i \phi_i}$$

$$\frac{\partial^2 \lambda_i}{\partial \gamma^2}(\Delta\gamma)^2 \cong \frac{1}{\psi_i \phi_i}\left[2\psi_i\{\Delta A_{sys} - \Delta\lambda_i^{1st} I\}\frac{\partial \phi_i}{\partial \gamma}\Delta\gamma\right]$$

Results: Accuracy & Speed (16 machine system)



Accuracy

- There are 6 critical contingencies.
- Algorithm identifies all 6 and 2 additional contingencies.

Speed

- Several times faster than full eigenvalue computation

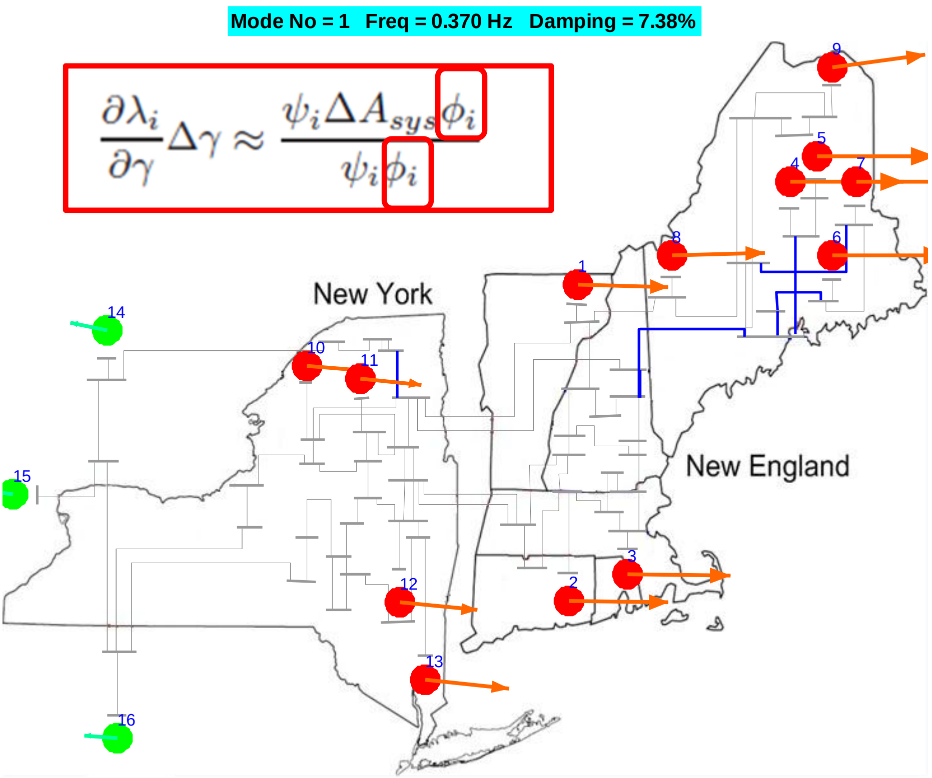
Next: How to resolve the violating contingencies during operation?

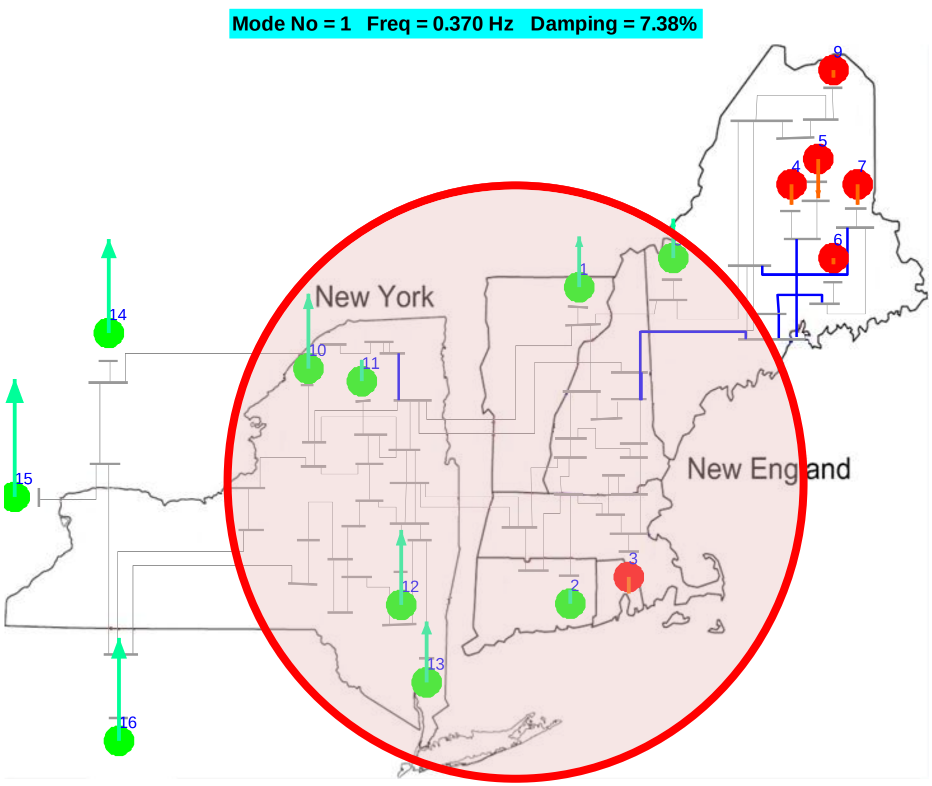
Oscillation Damping Control

Re-dispatch based damping control: Key Questions

- Which generators should increase output and which ones should decrease output?
- Which generators will be more effective in impacting a certain mode?
- What is the optimum amount of generation re-dispatch needed to damp the oscillation?
- How to avoid negative interaction between multiple modes?

Mode Shape vs Sensitivities





Mode-shape Grouping

Eigen-sensitivity Grouping

Re-dispatch ensuring post-contingency stability

- Objective: Achieve a minimum settling time for post contingency condition after re-dispatch
- Step I: Compute the sensitivities $\frac{\Delta\sigma}{\Delta \mathrm{P}}$ and $\frac{\Delta\omega}{\Delta \mathrm{P}}$ for each generator under post-worst case contingency
- Step II: First, the targeted *change* in the real part is determined by the equation $\Delta\sigma_{target} = \sigma_{post-target} - \sigma_{post}$
  - Next the target of the base case sigma under post-dispatch condition is determined using the value of $\Delta\sigma_{target}$ as $\sigma_{target} = \Delta\sigma_{target} + \sigma_{base}$
- Step III: Evaluate the dispatch command by QP to achieve σ_{target} with constraints
- Step IV: Run time-domain simulations for a) under base case b) most critical contingency conditions.

Accomplishments

Accomplishments

- Parallelization implementation in existing PSLF structure using (~15% speed gain with 2 cores)
- Compared several linear solver implemented the best performing solver in PSLF. 30 times factorization speed improvement.
- Developed implicit integration method with time-stacking approach and preliminary implementation with classical generator model.
- Developed new approach for fast contingency screening for small signal stability (IEEE PES GM 2014 Paper)
- Developed new approach for oscillation damping control

Project Team

Project Team

Dr. Naresh Acharya
Principal Investigator
GE Global Research

GE Global Research
- Dr. Nilanjan Ray Chaudhuri
- Dr. Chaitanya Baone
- Dr. Santosh Veda

GE Energy Consulting
- Dr. Juan Sanchez-Jasca
- Dr. Bruno Leonardi

PNNL
- Dr. Henry Huang
- Dr. Ruisheng Dao
- Dr. Di Wu
- Dr. Shuangshuang Jin
- Dr. Yousu Chen
- Dr. Yu Zhang

SCE
- Dr. Nagy Y. Abed
- Dr. Farrokh Habibi-Ashrafi

Acknowledgement

Acknowledgement: *"This material is based upon work supported by the Department of Energy under Award Number DE-OE0000626"*

Questions?

acharyan@ge.com