# Power System Transient Stability Simulation


**Ashley Cliff**

**Mentors:**

**Srdjan Simunovic**

**Aleksandar Dimitrovski**

**Kwai Wong**


JICS Joint Institute for Computational Sciences

NSF

Oak Ridge National Laboratory

The University of Tennessee Knoxville

# Goal

- **Overall Goal: Determine how to stabilize the system before it collapses by running simulations of initial causes faster than real time by implementing the Parareal Algorithm**

- **Personal Goal: Parallelize the MATLAB code and determine speed up**

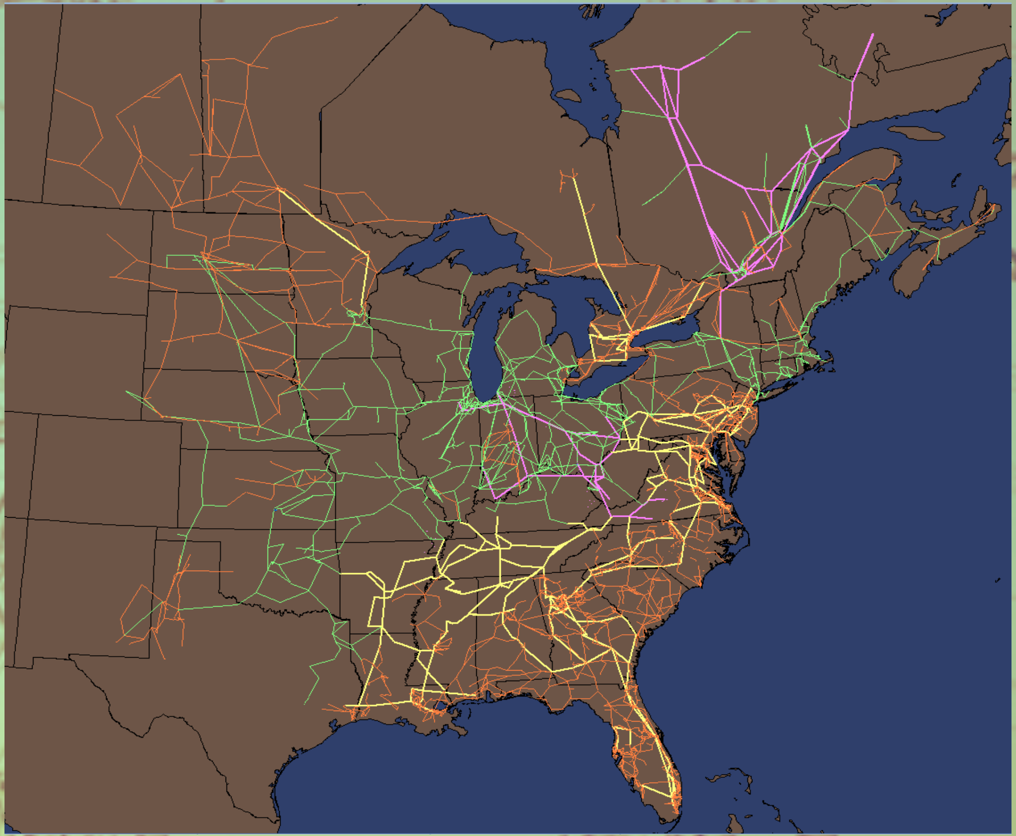
# Background

- Power System: Power lines with transformers, buses, generators, loads, etc.

- Interconnected Systems: East, West, Texas

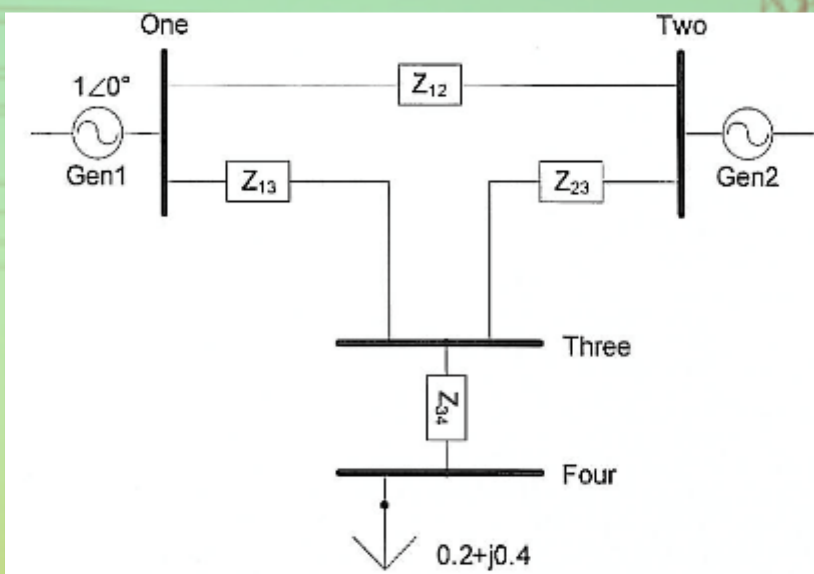
- High Performance Computing becomes necessary

- Power Failures: Creation and avoidance



# Steady State System Simulation

- **Determine voltage necessary to keep system at equilibrium**
- **Load amounts given, flat start (zero generation)**
- **Admittance Matrix** *Y*= [■■(*y*↓11 +*y*↓13 )&−*y*↓12 @−*y*↓12 &(*y*↓12 +*y*↓23 ) &■−*y*↓13 & 0@−*y*↓23 & 0 @■ −*y*↓13 & −*y*↓23 @0& 0 &■(*y*↓13 +*y*↓23 +*y*↓34 )& ]



# Steady State Solution

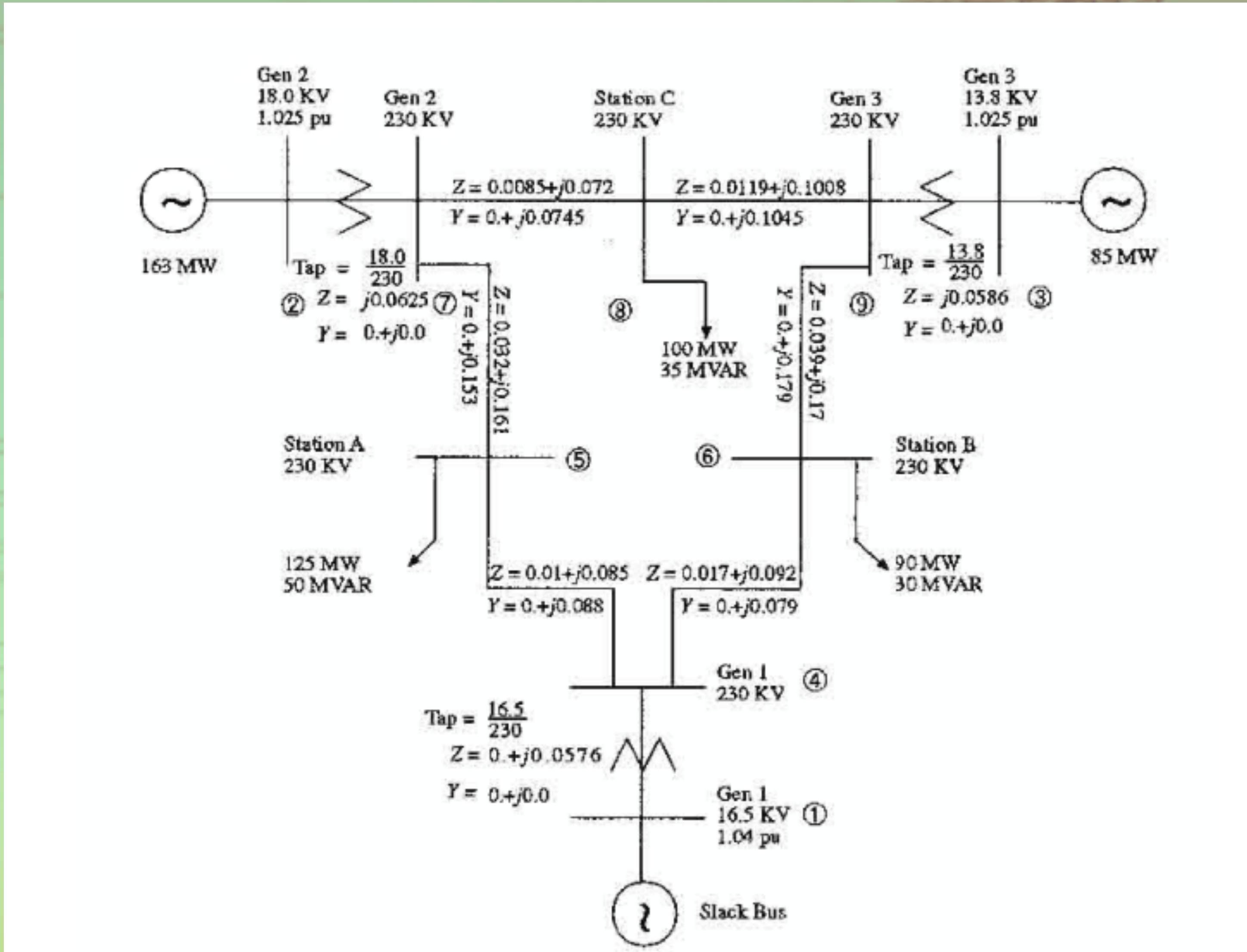
- **Load buses (PQ), Generator Buses (PV), Slack Bus**
- **MatPower Solver, Newton's Method**
- **Real and Imaginary Power:**

$$P_i^{sp} = P_i(\theta, V) = V_i \sum_{k=1}^{n} V_k \left( G_{ik} \sin\theta_{ik} + B_{ik} \cos\theta_{ik} \right)$$

$$Q_i^{sp} = Q_i(\theta, V) = V_i \sum_{k=1}^{n} V_k \left( G_{ik} \cos\theta_{ik} + B_{ik} \sin\theta_{ik} \right)$$

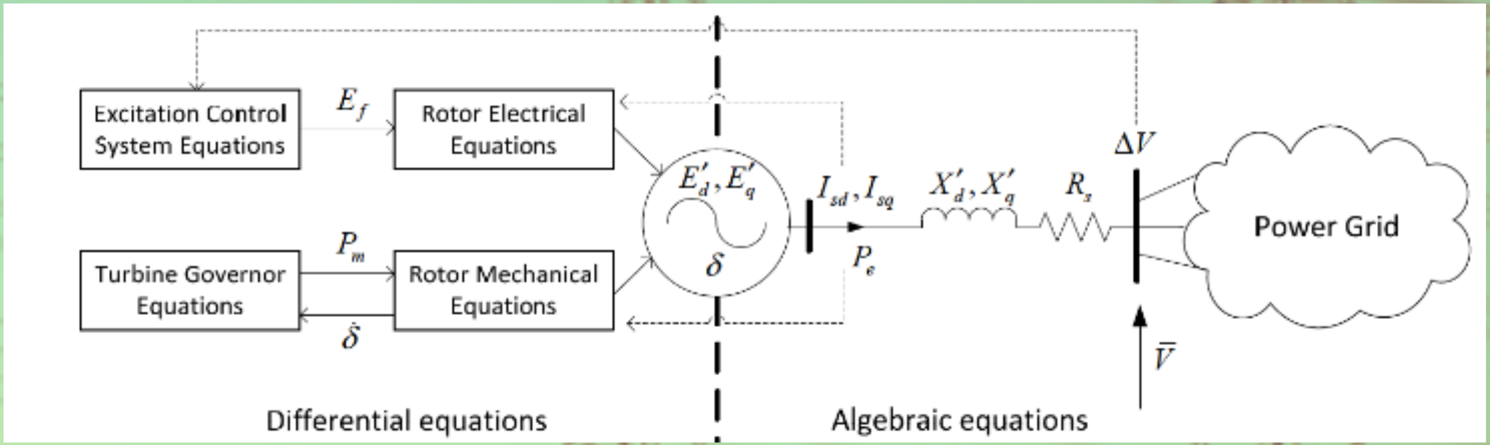
- **Solve for voltages and voltage angles**

# 3 Generator 9 Bus Diagram



# Dynamic System Simulation

- **Initial Conditions: Steady State Values**
- **Create a fault: Downed line, generator outage, etc.**

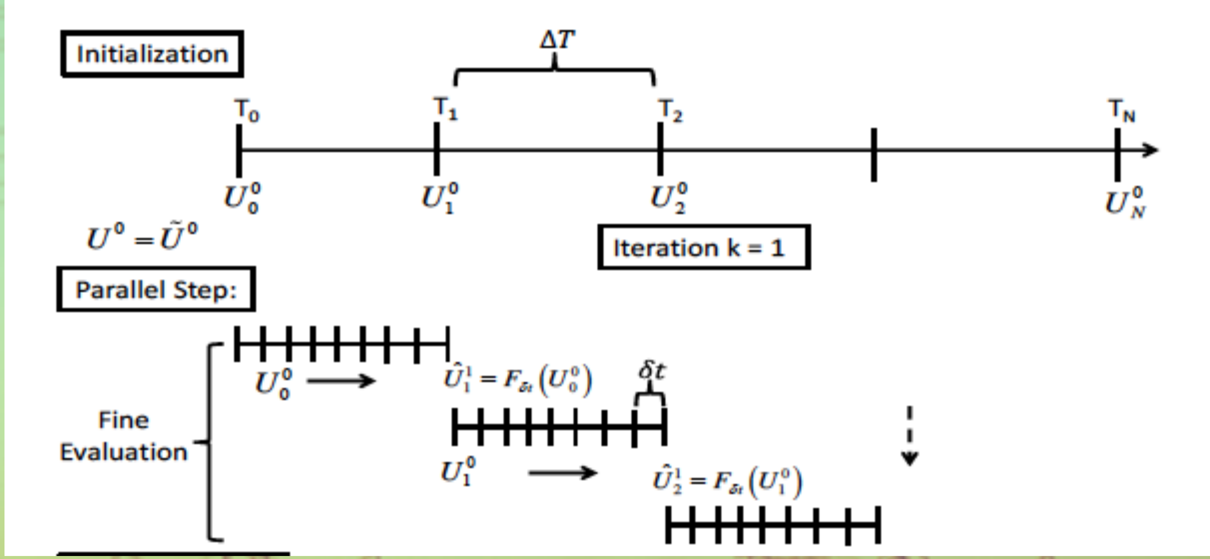



- **Solve Differential and Algebraic Equations**

| Name | Equation | MATLAB Function |
|---|---|---|
| Stator Algebraic Equation | $\begin{bmatrix} i_q \\ i_d \end{bmatrix} = \frac{1}{R_a^2 + X_d X_q} \begin{bmatrix} R_a & X_d \\ -X_q & R_a \end{bmatrix} \begin{bmatrix} E_q - V_q \\ E_d - V_q \end{bmatrix}$ | Eq_StatorAlgebraic22.m |
| Network Algebraic Equations | $I^{DQ} = Y^{DQ} V^{DQ}$<br>$Y_{ij}^{DQ} = \begin{bmatrix} B_{ij} & G_{ij} \\ G_{ij} & -B_{ij} \end{bmatrix}$;<br>$V_j^{DQ} = \begin{bmatrix} V_{Qj} \\ V_{Dj} \end{bmatrix}$; $I_i^{DQ} = \begin{bmatrix} I_{Di} \\ I_{Qi} \end{bmatrix}$; | NWAlgebraic22.m |
| Governor Model | $\frac{dP_{SV}}{dt} = \frac{1}{T_{SV}} \left[ -P_{SV} + P_C - \frac{1}{R_D} S_m \right]$ | Eq_SteamGov.m |
| Turbine Model | $\frac{dT_m}{dt} = \frac{1}{T_{CH}} \left[ -T_m + P_{SV} \right]$ | Eq_SteamTurb.m |
| Change in q - axis Transient Voltage | $\frac{dE'_q}{dt} = \frac{1}{T'_{do}} \left[ -E'_q + (X_d - X'_d) I_d + E_{fd} \right]$ | Eq_ExcType1.m |
| Change in d- axis Transient Voltage | $\frac{dE'_d}{dt} = \frac{1}{T'_{qo}} \left[ -E'_d - (X_q - X'_q) I_q \right]$ | Eq_ExcType1.m |
| Change in Exciter Field Voltage | $\frac{dE_{fd}}{dt} = \frac{1}{T_E} \left[ -(K_E + A_E (e^{B_E E_{fd}})) E_{fd} + V_R \right]$ | Eq_ExcType1.m |
| Change in Rotor Angle | $\frac{d\delta}{dt} = w_B S_m$ | Eq_Gen22.m |
| Change in Slip | $\frac{dS_m}{dt} = \frac{1}{2H} \left[ -D S_m + T_m - T_e \right]$ | Eq_Gen22.m |
| | $\frac{dE_{dc}}{dt} = \frac{1}{T_c} [ -E_{dc} - (X'_q - X'_d)$ | |

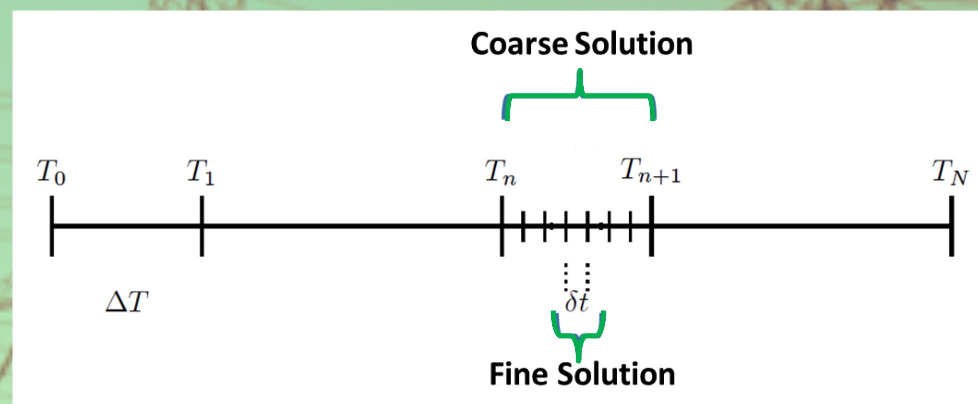
# Parareal in Time Algorithm

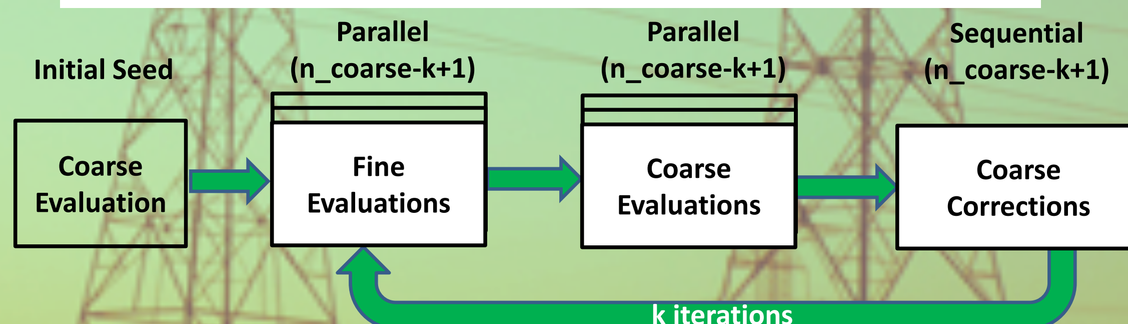
- Divides the time domain into intervals, and integrates concurrently over each interval.
- Used rather than spatial decomposition methods
- Coarse solve then fine solve in parallel



# Methodology

- Coarse solution – Trapezoidal Rule (RK2)
- Fine solution - Runge-Kutta 4 method
  - Solve differential equations and algebraic equations in a predictor-corrector approach
- k1 – k4 equations



# Pseudocode

**Trapezoid Function Call – Initial coarse evaluation**
**While iterations less than max number of iterations:**

- **For each coarse section (in parallel):**
  - **Runge-Kutta 4 Function Calls - fine evaluation**
- **Correct coarse evaluation**
- **Add one to iteration count**

**MATLAB 'parfor' tests**

| Matrix Size | Number of Loops | Serial Time | Parallel Time |
|---|---|---|---|
| 10,000 | 32 | 84.986s | 119.811s |
| 10,000 | 64 | 267.242s | 232.232s |
| 5,000 | 1024 | 401s | 314s |
| 5,000 | 512 | 236s | 262s |

# Results

- Theoretical Speed up with 32 workers and 6 iterations:

  32/6 ~ 5.33

  - Where N is the number of parallel iterations running, k is the number of iterations to converge and N/k is the speed up

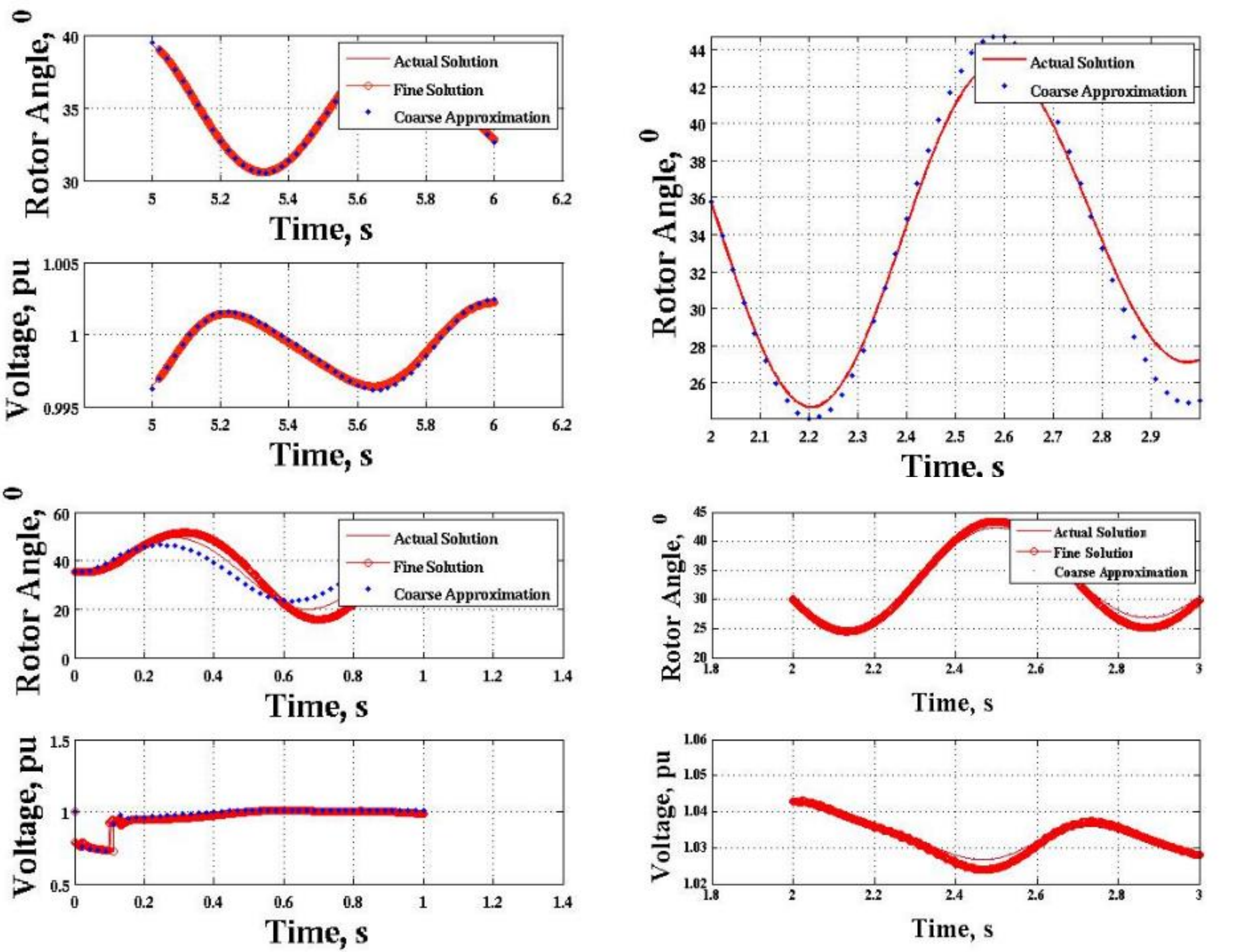
- Measured Speed up with 32 workers:

  5.423s/1.151s ~ 4.7

  - Where the numerator is the time for the serial loop to execute all iterations and the denominator is the time for the parallel loop to execute all iterations

- MATLAB parallelization overhead costs

- Conceptual Success



Rotor Angle, 0
Time, s
Actual Solution
Fine Solution
Coarse Approximation
Voltage, pu
Time, s
Rotor Angle, 0
Time. s
Actual Solution
Coarse Approximation
Rotor Angle, 0
Time, s
Actual Solution
Fine Solution
Coarse Approximation
Rotor Angle, 0
Time, s
Actual Solution
Fine Solution
Coarse Approximation
Voltage, pu
Time, s
Voltage, pu
Time, s

# Conclusion/Future Work

- Steady state to dynamic system
- Parareal Algorithm implementation
- Personal goal accomplished: Parallelized MATLAB code and conceptually proved the speed up

- Optimize the program
- Parallelize the fault cases
- Benchmarking
- Create C/C++ Version

# Sources


- Gurrala, Gurunath. "Power System Parallel Dynamic Simulation Framework for Real-Time Wide-Area Protection and Control."

- Gurrala, Gurunath, Aleksandar Dimitrovski, Pannala Sreekanth, Srdjan Simunovic, and Michael Starke. "Parareal in Time for Fast Power System Dynamic Simulations."

- Meier, Alexandra Von. *Electric Power Systems A Conceptual Introduction*. Hoboken, N.J.: IEEE :, 2006. Print.


Questions?