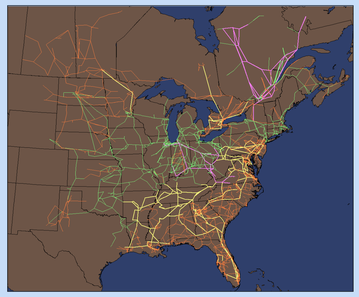
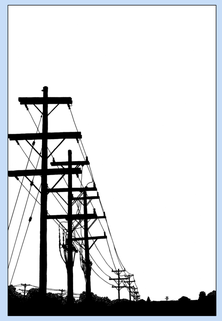
Power System Transient Stability Simulation

Ashley Cliff, Central College
Mentors: Srdjan Simunovic, Aleksandar Dimitrovski, ORNL; Kwai Wong, UTK

Abstract

The purpose of this project is to create accurate simulations of power outages that can be used to shorten the duration and number of occurrences of power failures. For the simulations to be useful, they must be able to run faster than real time, to determine what will happen when there's an outage before the outcome occurs. An innovative way to achieve that speed up is with the Parareal Algorithm.





Steady State System

- Basis for Dynamic Systems
- Determine voltage and voltage angles
- Match real power and imaginary power generation to consumption
- Solved using Newton's Method

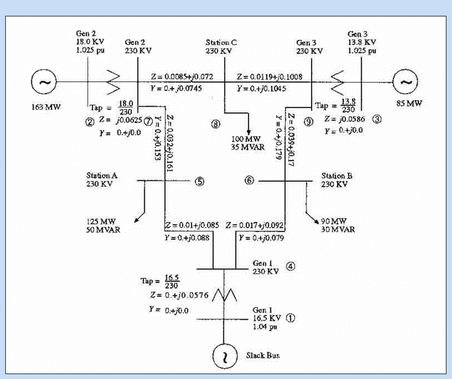
$$P_i^{sp} = P_i(\theta, V) = V_i \sum_{k=1}^{n} V_k(G_{ik}sin\theta_{ik} + B_{ik}cos\theta_{ik})$$

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

Coupled non-linear algebraic equations

$$Y = \begin{bmatrix} (y_{12}+y_{13}) & -y_{12} & -y_{13} & 0 \\ -y_{12} & (y_{12}+y_{23}) & -y_{23} & 0 \\ -y_{13} & -y_{23} & (y_{13}+y_{23}+y_{34}) & -y_{34} \\ 0 & 0 & -y_{34} & y_{34} \end{bmatrix}$$

Admittance Matrix

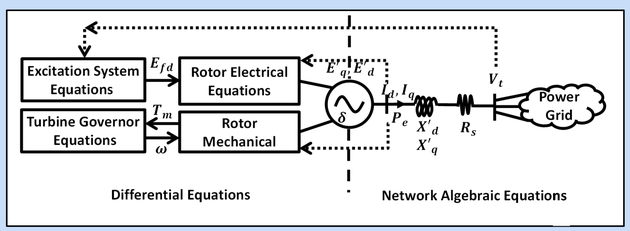


3 Generator, 9 Bus System

The MatPower program takes an admittance matrix created from a bus diagram as input and solves the power equations for the state variables.

Dynamic System

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



Parareal Implementation

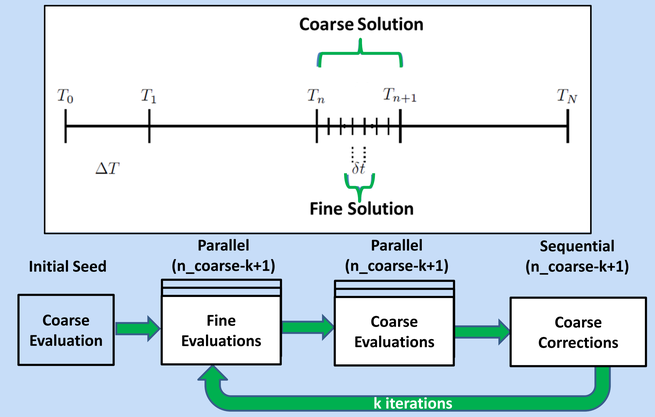
Once a fault has tripped, the final solution for the steady state system is used as the initial values for the dynamic system problem. The goal is to accurately simulate how the fault propagates through the system as time goes on.

The RK4 method is then used to determine the state values for the next iteration. This process is repeated until the error is within a designated margin or the max number of iterations is reached.

Using Parareal: Time sections can run at the same time, with a coarse approximation used to generate initial values for each iteration

Parareal Concept

The Parareal in Time Algorithm divides the time domain into intervals, and integrates concurrently over each interval.



MATLAB 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                    Fine solve loop
```

Results

Based on the analysis of the fine solve loop for multiple fault cases for the 3 generator 9 bus system with 32 workers.

Theoretical Speed up – number of sections (or number of workers) divided by the number of iterations
Actual Speed up – total time for serial loop to run divided by total time for parallel loop to run

Theoretical Speed Up: 32 workers/6 iterations = 5.3
Actual Speed up: 5.423s/1.151s = 4.71

Conceptually, parallelizing the loop does create a speed up. In the future, to increase the speed up for the entire program, it will be written in C or C++, which can be optimized better. MATLAB has a high set up cost (in time) to run in parallel.

Acknowledgements

Thanks to NSF, University of Tennessee Knoxville, JICS, and Oak Ridge National Lab.