System Under Investigation Power systems are large and complicated structures composed of many connected objects. Each power system object has a functional purpose and reacts to various changes differently. These objects can be modeled mathematically as transfer functions that alter certain parameters (states) of the objects themselves, or the system as a whole, over time.

Software exists to model transient (fast) reactions of power systems. These involve small time steps so that models can have fast time constants and capture the oscillatory nature of power system responses. While these models have their place, it may be desirable to calculate slower, more long term dynamic responses to gradual perturbances such as wind ramps, daily load cycles, or balancing authority type situations.

PSLF is a commercial power systems analysis software package by GE that is regarded as an industry standard. Many working and accepted system models exist that are compatible with PSLF. Additionally, GE has enabled other code languages to interface with their software for a more customized experimentation and simulation process. However, this code support is not completely functional (dynamic simulation is not fully supported).

The first goal of this project is to design a simulation framework that performs long term dynamic (LTD) simulations using available PSLF systems, dynamic data, and modified models. An Agent Based modeling approach will be taken to accommodate a wide variety of power systems and enable the independent development of features. The specialized functionality of the framework will attempt to facilitate investigating system reactions that may have been previously overlooked, difficult, or computationally heavy, to simulate.

When performing such LTD simulations, certain simplifying assumptions can be made:

1. The system is stable (synchronized)
2. Because of the synchronized assumption, the system has only 1 frequency and is altered by the aggregate swing equation

$$\dot{\omega}_{sys} = \frac{1}{2H_{sys}} \left(\frac{P_{acc,sys}}{\omega_{sys}(t)} - D_{sys} \Delta\omega_{sys}(t) \right)$$

3. Time steps of 1 second (or larger) will allow for ignoring transient rotor windings and generator damping time constants.
4. Dyanmic models that have time constants smaller than the system time step (1 second) must be reworked to avoid mathematical errors.
5. Accelerating power will be distributed to system generators participating in the inertial response according to a ratio of object inertias.

The specific software used will be PSLF (for powerflow calculations), the PSLF .NET API, and ironpython. This choice is due to the alleged trustworthiness of the systems modeled in PSLF and the future possibility of a python API. Additionally, long term simulations my be performed through PSLF to confirm LTD simulation results. It is believed that the custom LTD code will perform the desired simulations much faster than PSLF alone and will exhibit reasonable accuracy. The python code shall allow for modular expansion and future development options.

Conceptual model

1. Software requirements: PSLF and middleware.dll, ironpython, and possibly visual studio for development.
2. The PSLF .sav and .dyd files contain data that describes the desired system.
3. Ironpython code will interact with PSLF to solve the .sav file power flow numerous times as system states change.
4. The python code will establish a 'mirror' of the PSLF system in python with data available from the solved system power flow, case parameters from the API, and dynamic model information parsed from the .dyd file.
5. A routine of agent steps will be developed that is capable of:
   (a) Calculating accelerating power (Pm-Pe-Perturbance)
   (b) Distributing electric power among generators in python 'mirror' according to machine inertia:
   $$P_{e,i} = P_{m,i}(t) - P_{acc,sys}(t)\frac{H_i}{H_{sys}}$$
   (c) Updating PSLF system with newly calculated generator powers
   (d) Performing power flow solution of PSLF system
   (e) Checking slack bus error adherence and repeating power distribution as required
   (f) Calculating system frequency response
   (g) Calculating all object dynamic responses
   (h) Logging states and increasing system time
6. After a simulation, output and system data shall be saved as a python binary object. This object could be processed into other data types for use in other software (MATLAB).
7. Quick display functionality may be built into the python mirror to view system activity immediately following execution.

Simulation Model Using an agent based approach, each object in the simulation will be used in various substeps per each time step. These steps will be carried out in a loop once the system has been initialized.
It is assumed that transfer function models may be simplified to state space arrays, or something equivalent, that can be populated by specific model parameters. Once solved, i.e. slope is found, other numerical techniques can be employed to predict the next system state.

Substeps : (Performed each timestep)

1. Perterubance step: A perturbance object will be programmed / configured before the simulation and attached to the environment. These agents will alter the power into/out of other system agents (such as generators and loads). Once this step is complete a total system Perturbance can be calculated.

2 a. Distribution Step: Once P_{acc} is calculated, it will be distributed to the system. These new values will be updated in the PSLF system.

2 b. Power Flow Solution : Using the PSLF powerflow solver allows the system to find balance. If slack error is too great, the error will be sent back to the beginning of the Distribution Step to be processed again. Note: Since this computation happens at the end of the Distribution Step, it may not be a unique step and more of a while: slack error > some error tolerance. Once the slack bus is below error, Electric power has been adequately distributed and will be sent to the Dynamics step as well as being summed for the next time step.

3. Dynamics step: Using current and history values, agent related models are simulated to calculate responses to the change in accelerating power and frequency. This will probably involve state space solutions to ODEs (transfer functions) and numerical integration (PSLF uses Adams-Bashforth). Additionally, prioritization may have to occur so that proper 'model chains' may be executed in a realistic manner.

Questions:
Should another powerflow be solved once dynamic responses have been recorded to PSLF / python mirror?

Agents: Each agent keeps track of its own history data and reference to PSLF Scanbus, Bus ID, and/or Bus Name. (still unsure of best method, all three references could be useful)

Core Agents

- Bus: load, generator, slack

  Keeps track of Voltage, angle, Pe, Q

  1. Usefuf for data collection
  2. All buses must act the same
  3. As each bus steps, it steps each attached Generator (or other LTD element)
  4. bus power and voltage will change after each individual element step.
  5. An `add_element` function will attach generators (or other bus altering agents) to the step array each bus processes. This execution array may involve prioritization if needed.

- Line: to keep track of line current
- Loads: For perturbance access and history
- Generators: Slack, non-slack
  1. Slack gen keeps track of expected Pe for error correction applications
  2. Will have a participation flag (default = 1)
  3. will have dynamic model list (possibly prioritized), that will execute each step
- Perturbance: to simulate known/possible events
  1. Applied to generators, loads, etc.
  2. steps, ramps, sines
  3. noise (adaptive or set range) - to add an element of randomness to simulation. Possibly un-necessary.

Additional Agents

- Balancing Authority
- Power Plant Supervisor
- Microgrid / Battery?

Model: Keeps track of global system parameters and dictates step and substep operations/sequence.

- time step
- system time
- simulation end time
- system H (either calculated via dyd or set by user input)
- system damping
- system history of: time, frequency, P_{acc} P_e P_m $P_{perturb}$

Additional Required Functions / Content:

- .dyd parser / PSLF dynamic model builder
- Dynamic go between
- LTD model library
- Data Collector / Exporter (MATLAB?)
- Data Presenter (grapher)

Current Tasks:

1. Identify Slack bus programatically.

   Possibly identify by generator with most busses under control? Proving more difficult than expected.

2. Further develop required elements for a simple 'swing only' run.

3. Numerical integration at beginning of simulation may require extra care as to avoid out of index errors. -Maybe not because python can do limited negative indexing.

4. Should there be another powerflow solve after the dynamic step so that the system can find balance before starting again? Seems like it should.

Recent Progress:

1. Set 'system islanding' off in solution parameter code.

   Accomplished by creating an `LTD_Solve` function inside the Model.
   To resolve missing .p file errors:

   (a) Extract `upslf.zip` (found in GE PSLF folder) in place.

   (b) remove the 'upfc' prepending from the non-found .p files. (as noted in terminal output)

2. Investigate line current data in PSLF:

   Seems as though line current isn't an attribute of a branch (line).
   Current could be calculated, but maybe it's available somewhere I'm un-aware of. Matt?

3. Initialization from PSLF CaseparDAO begun.

4. Investigate the impact of ignoring armature losses:

   Ra set to zero in all mini/microWecc genrou models...
   Is this right? (file from RJ's old computer)