Introduction: 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 are often modeled mathematically as transfer functions or state equations that alter certain parameters (states) of the objects themselves, or the system as a whole, over time.

A wide variety of software exists to model transient (fast) reactions of power systems. These softwares 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, or disturbances, such as wind ramps, daily load cycles, or balancing authority type situations.

General Electric's Positive Sequence Load Flow (PSLF) is a commercial power systems analysis software package that is regarded as an industry standard. Many working and accepted system models exist that are compatible with PSLF. Additionally, GE has created a .NET interface to their software for a more customized experimentation and simulation process.

Goals: The first goal of this project is to develop software that performs long-term dynamic (LTD) simulations using available PSLF systems, dynamic data, and modified/custom dynamic models. An Agent Based modeling approach will be taken to accommodate a wide variety of power systems and enable the independent and modular development of features.

Once complete, the specialized functionality of the software framework will attempt to facilitate investigating system reactions that may have been previously overlooked, computationally heavy, or programmatically difficult to simulate via other methods.

System Under Investigation: When performing such LTD simulations on multi-machine power systems, certain simplifying assumptions will be made:

1. The system is synchronized.
2. The system has only 1 frequency and it is altered by the aggregate PU swing equation

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

3. System damping D_{sys} is ignored (set to 0).
4. Time steps of 1 second (or larger) will allow for ignoring transient rotor windings and generator damping time constants.
5. Existing dynamic models with time constants smaller than the system time step (1 second) must be reworked to avoid mathematical errors.
6. Accelerating power will be distributed to system generators participating in the inertial response according to a ratio of object inertias.

Conceptual Model: Ironpython (IPY) will take file locations of PSLF .sav (power system information file) and .dyd (PSLF dynamic model information) files that will be used to generate a python 'mirror' of a PSLF power system. IPY will communicate to PSLF via the .NET API to gather and/or update system data (states) and run PSLF power-flow calculations. Between power-flow calculations, custom dynamic models will react to the most recent power-flow data to change associated IPY 'mirror' and PSLF states. Finally IPY will save/export collected data and exports a .mat file by running a Python 3 script.

Long-term simulations my be performed by the GE dynamic simulation (PSDS) to validate LTD simulation results. It is believed that the custom LTD code will perform the desired simulations much faster than PSDS and will exhibit reasonable accuracy to solve engineering problems.

Conceptual Model Details of Note:

1. Software requirements: PSLF with middleware.dll, Ironpython, Python 3, and Visual Studio with Python Development Tools (plus GitHub extension).
2. The PSLF .sav and .dyd files contain data that describes any given system.
3. Ironpython code will establish 'mirror' of the PSLF system with data available from solved PSLF case data, system parameters from the API, and dynamic model information parsed from the .dyd file.
4. Ironpython code will interact with PSLF to solve PSLF power flow numerous times and record data as system states change.
5. A routine of Model and Agent steps will be developed that is capable of:
   (a) Calculating system frequency response from any remaining system Pacc
   (b) Calculating current accelerating power (Pm-Pe-Perturbance) where:
      i. Pm will be collected from generators in the system
      ii. Pe will be a summation of the previous time steps solved power flow
      iii. Perturbance data will be summed per time step from all perturbance agents.
   (c) Distributing electric power among generators in python 'mirror' according to machine inertia:
   $$P_{e,i}(t) = P_{e,i}(t-1) - \Delta P_{acc,sys}(t)\frac{H_i}{H_{sys}}$$
   Note that this will only affect generators participating in the inertial response.
   (d) Updating PSLF system with newly calculated generator powers and voltages.
   (e) Performing PSLF power flow solution of system.
   (f) Checking slack bus error and repeating power distribution as required while also ensuring convergence.
   (g) Calculating object dynamic responses in a prioritized manner.
   (h) Logging states and increasing system time.
6. After a simulation, desired data may be saved as a pickled python object, dictionary, or MATLAB .mat file.
7. Quick display functionality may be built into the python mirror to view system activity immediately following execution using the `matplotlib.pyplot` library.

Simulation Model: Using an agent based approach, each object in the simulation will be activated in various substeps per each time step. These steps will be carried out in a loop once the system has been initialized for a predetermined amount of time / time steps unless an error occurs or the system fails to converge.

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. Alternatively, Ironpython, or available .NET, ode solvers may be used if applicable and available.

Substeps : (Performed each time step/ main model step)

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 by non-Slack generators 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, a different method may have to be used due to timestep size). Additionally, prioritization may have to occur so that proper 'model chains' may be executed in a realistic manner.

Agents: Each agent keeps track of its own history data and references to PSLF. PSLF references are numerous and may only apply to certain agents. Some examples are: id, longId, Area, Zone, Bus number, Bus name, Scan bus number, External Number

Core Agents

- Bus: Keeps track of Voltage Magnitude and Angle
    1. Useful for data collection
    2. As each bus steps, it steps each attached Generator (or other LTD element)
    3. bus voltage and angle will change after each individual element step.
- Line / Branch: to keep track of line current
- Loads: For perturbance access and history of P, Q, and status.
- Generators: Slack, non-slack
    1. Track Pe, Pm, Q, and status.
    2. Slack gen keeps track of expected Pe for error correction applications
    3. Will have a participation flag (default = 1)
    4. will have dynamic model list (probably prioritized), that will execute each step
- Perturbance: to simulate known/possible events
    1. Various Types: steps, ramps, sines
    2. Applied to generators, loads, or other controllable element (breakers?).
    3. Control of P, Q, V, status
    4. Additional type: 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
- Area
- Shunt & SVD

Model: Initializes PSLF, mirror, and agents. Keeps track of global system parameters and dictates step and substep operations/sequence.

- Required Inputs:
    1. File locations for PSLF
    2. time step
    3. simulation end time
    4. Slack error tolerance
- Optional Inputs:
    1. H of system (else calculated)
    2. D of system (else calculated)
    3. Debug flag
- Keeps current value and hisotry of:
    1. time
    2. system frequency
    3. power summations: P_{acc} P_e P_m $P_{perturb}$

Additional Required Functions / Content:

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