Recent Progress:

1. Task: Set area interchange off in solution parameter code.

   Accomplished by creating an `LTD_Solve` function inside the Model.
   Missing .p file errors resolved by setting gcdAdjustment = 0

2. Task: Develop required elements for a simple 'swing only' run including:

   - Simple dyd parser
   - H linking system
   - Load perturbance agent
   - Pacc distribution function
   - Combined swing equation (unverified correctness)
   - logStep for Mirror, Bus, Load, and Generators

3. GitHub repository updated:
   See: `https://github.com/thadhaines/LTD_sim`

Current Tasks:

1. Verify Swing only run results with known PSLF results.

2. Develop better data output functionality.

   Will probably include a csv output, a matlab & python csv reader script, and basic plotting templates/functions

3. Initialization of mirror from PSLF system continued:

   Shunts and SVDs have yet to be modeled. They are required to accurately calculate system Q losses.

4. Investigate line current data in PSLF:

   A FlowtabrDAO exists that can find flow between busses. A way to initialize bus connections between areas has yet to be devised.

5. Identify Slack bus programatically. (No Progress since last time)

   Can locate when only 1 Slack exists. If more than one Slack, maybe identify by generator with most busses under control? Proving more difficult than expected. Can identify in PSLF via the `summ()` or `isld()` commands.

Current Questions:

1. Next step after verification of current progress to model the `genrou` or `gensal`?

2. Is there any available/relevant event data that may help us to verify simulations of specific instances (wind ramps or other behavior) that the novel research will focus on? (Same as last time)

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 at this time (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 dynamic 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

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

3. Time steps of 1 second (or larger) will allow for ignoring transient rotor windings and generator damping time constants.
4. Dynamic 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 verify 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 Visual Studio with Python Development Tools.

2. The PSLF .sav and .dyd files contain data that describes any given system.

3. 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.

4. Ironpython code will interact with PSLF to solve the .sav file power flow numerous times and record data as system states change.

5. A routine of agent steps will be developed that is capable of:

   (a) Calculating 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.

   (b) Distributing electric power among generators in python 'mirror' according to machine inertia:

   $$P_{e,i} = P_{e,i}(t) - \Delta P_{acc,sys}(t)\frac{H_i}{H_{sys}}$$

   Note that this will only affect generators participating in the inertial response.

   (c) Updating PSLF system with newly calculated generator powers and voltages.

   (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 in a prioritized manner.

   (h) Possibly performing one more power flow so system is balanced after mechanical response?

   (i) Logging states and increasing system time.

6. After a simulation, desired 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 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.
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, python ode solvers may be used if applicable.

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). 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)