Recent Progress:

1. Global g, linear & non-linear functionality added for:
   - system variables
   - non-conforming loads
   - simulation control
   - PSS
2. ODE test for variable time step
3. Created and tested `pssGainFix` for pss model differences between version 2 and 3 based on user manual and Ryan info.
4. Fixed corrector integration in s_simu_Batch of dpw states
5. Created more MATLAB plot functions to compare PST data
6. GitHub updated:
   `https://github.com/thadhaines/MT-Tech-SETO`

Sandia Action Items:

- Continue development of pwrmod / ivmmod models and their implementation in PST.
- Decide on PST base version (3.1⟶SETO)
- Plan for variable time step methods
- Investigate power electronics-based models (REGC - Matt)

Current Questions:

1. Requirements for variable step methods:
   - Functionalized Network solution
   - Functionalized Derivative calculation
   - Functionalized collection and return of calculated derivatives
   - 'outputfunction' that handles logging of correct output values and indexing
   - updated scheduler to run simulation in 'blocks' between known events.
2. PST modeling of transformers?
3. Play in data for variable solar irradiance?
4. Deadlines of any sort?

Current Tasks:

1. Remaining model globals to structure:
   - IVM
   - SVC
   - TCSC
   - HVDC
   - pss design
   - Induction Motor
   - Induction Generator
   - delta P omega filter
2. Test models in non-linear and linear simulations
3. Explore/Create example cases
4. Get code/cases from Ryan
5. Think about AGC implementation.
   - Create `area_con` and `agc_con`
   - Calculate interchanged line flows each timestep
   - Calculate Area/System Average Frequency
   - Distributed ACE ⟶ a step to governor pref via `mtg_sig`?
6. Think about using standard ODE solvers
7. Think about cleaning up or flowcharting svm_mgen_Batch
8. Work on understanding PST operation
9. Document findings of PST functionality
10. Investigate Octave compatibility

Coding Thoughts:

1. Condense ≈340 globals into 1 structured array with ≈18 fields based on category.
2. Create new `s_simu_Batch` style script that functionalizes the newtork and dynamic calculations so that standard MATLAB ODE solvers may be used.
3. Rework how switching & perturbance events are handled into a more flexible and general format. (use flags)
4. Generate something similar to unit test cases to verify code changes don't break everything during refactor.
5. Generate comparison scripts to verify simulated results match after code changes.