Recent Progress:

1. Global g,non-linear functionality added for:
   - interface values (Y matricies)
   - line
   - bus
   - areas
   - line monitor (lmon)
   - AGC
2. Fixed limiting issue in lmod/rlmod.
3. Created line monitor, area and AGC models/functionality.
4. Created AGC example
5. Lightly documented AGC example
6. Updated pst SETO change doc
7. 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. Differences in `mac_ind` between versions.
2. Induction machines have no speed? only angle?
3. PST modeling of transformers?
4. Play in data for variable solar irradiance?
5. PSS design doesn't seem to be used in normal simulation?
6. Deadlines of any sort?
7. Continued employment beyond August 12th?

Current Tasks:

1. Think about using standard ODE solvers
2. 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.
3. Decisions concerning remaining globals:
   - IVM (waiting for linear code)
   - delta P omega filter (no examples)
   - PWR (cell data only)
   - pss design (not used in simulation)
4. Refine AGC implementation.
   - update/create pstSETO flowchart
   - create algorithm flowchart
   - Add conditional ACE
   - Account for tripped gens H removal.
5. Think about cleaning up or flowcharting svm_mgen_Batch
6. Work on understanding PST operation
7. Document findings of PST functionality
8. 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. (flags? objects?)
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.