Recent Progress:

1. Added Exciter models 0-4 to global g, tested as functional in both linear and non-linear batch runs.
2. Added Syncrhonous machine models em, sub, tra, and sub_NEW to global g, tested as functional in both linear and non-linear batch runs.
3. Conceptual work on `lmon` - a line flow montior for non-linear simulation
4. Collected thoughts on AGC.
5. GitHub updated: `https://github.com/thadhaines/MT-Tech-SETO`

Sandia Action Items:

- Continue development of current injection and voltage injection converter models and their implementation in PST.
- Be aware of multi rate methods
- Plan for variable time step methods
- Investigate power electronics-based models

Current Tasks:

1. Continue to cast globals to structured g
2. Test models in both non-linear and linear simulations
3. think about cleaning up svm_mgen_Batch
4. Explore/Create v3 example cases
5. Think about AGC implementation.
6. Reference Trudnowski code for 'structured array, functionalized' approach.
7. Reference Stajcar code for basic transient stability simulation flow.
8. Work on understanding PST operation
9. Document findings of PST functionality
10. Investigate Octave compatibility

Possible Future Tasks:

1. Investigate Sandia integrator stability methods. See if the modified PST used by Sandia in 2015 paper exists for an example of how they implemented different integration routines / stability calculations. (Contact Ryan?)

Coding Thoughts:

1. Condense ≈340 globals into 1 structured array with ≈18 fields based on category that contain PST arrays used for vector calculations. ex: `g.lmod.lmod_con`, `g.sys.t`
2. Enable 'objects' (structure of arrays), but include functions to interact with condensed globals so vectorized operations are still possible.
   This requires more conceptual modeling to understand what needs to be passed/references/changed for each 'object'. Would enable addition of area definitions to models.
3. Separate total system calculation of derivatives into scripts/functions to allow for easier changing of integration method. Possibly employ `feval` for a more dynamic calculation routine.
4. Rework how switching & perturbance events are handled into a more flexible and general format. (use flags)
5. Generate something similar to unit test cases to verify code changes don't break everything during refactor.
6. Generate comparison scripts to verify simulated results match after code changes.

Current Questions:

1. Requirements for variable step methods
2. PST modeling of transformers?