# SBvar: Varying Parameters Analysis for Systems Biology

Component Specification

Rachel Ng

BIOEN 537 Software Project

## Software Components

### <u>Modeling</u>

#### Experiments (OneWayExperiment, TwoWayExperiment)

Series of classes for handling an object that stores the tellurium model, factors, and levels. It will store the list of parameter conditions and their associated simulation results.

Input and Attributes:

- Model specifications: Antimony model, selection of variables to simulate and calculate steady states.
- Condition specifications: Name of factor(s) to be varied, range of values to use
- Simulation parameters: start, end, and number of timepoints

Attributes:

- Steady states: 2D array storing steady states across all conditions
- Simulations: High dimensional array storing time series across all conditions
- Obs: Dataframe with conditions along its index for storing annotations/labels.

Methods:

- **set_conditions**
  Generates all 1-dimensional or 2-dimensional grids of experimental conditions based on user input and model specifications. Conditions are stored as Experiment attributes.
- **simulate**
  Iterate through each experimental condition and perform simulation.
- **calc_steady_state**
  Iterate through each experimental condition and calculate steady state
- **get_values/get_mesh**
  Function for getting vector or mesh grids corresponding to steady state value or time point value for a variable across all conditions. Also used for getting mesh grid representations of varying parameters and time points.
- **plot_line (OneWay Experiment)**
  Based on user selection, plots line plot illustrating relationship between steady state, simulation values and one varying parameter.
- **plot_mesh (TwoWay Experiment)**
  Based on user selection, get mesh grids and generates 2D or 3D plots illustrating relationship between steady state, simulation values and two varying parameter
- **plot_timecourse_mesh (OneWay Experiment)**
  Plot 3D plot of time courses of one response variable across multiple levels of a factor.

## Data Transformation

Utility functions for transforming vector of values associated with each condition into 2D mesh grids that have Cartesian indices with specified variables varying along the x- and y-axes.

**meshes_to_meshvector**
Transform list of 2D mesh grids into a tidy array where each row corresponds to one condition and columns are the varying parameters.

**vector_to_mesh**
Transforms one dimensional vector into mesh grid.

**meshvector_to_meshes**
Reverses meshes_to_meshvector. Transform a tidy array of conditions into mesh grids, one for each parameter.

## Time course analysis

**cluster_timeseries**
Cluster time courses for one response variable across simulations using k-means clustering.
Input: Takes an experiment object and selection of variable.
Output: Array of cluster labels (also added to obs annotation dataframe in experiment object)
Available Methods: k-means clustering (Future implementations: k-shape, and dbscan)

**fit_sinusoid (incomplete)**
Fit Sinusoid to a time course for a selection of variable for each condition.
Input: Takes an experiment object and selection of variable.
Output: Array of oscillation metrics (added to obs annotation dataframe in experiment object)

**find_max (incomplete)**
Find first time point the time course reaches its global maximum for each simulation
Input: Takes an experiment object and selection of variable. Take additional tolerance argument to search for first time point within some percent of global maximum.
Output: Array of global maximum (added to obs annotation dataframe in experiment object)

## Plotting

**plot_mesh**
Plot 2D contour plot or 3D surface plot from mesh grid data.
Input:
- Data: Mesh grids for the x, y, and z variable.
- Kind: filled contour, contour, or surface plot
- Projection: 2D or 3D projection

Output: matplotlib figure and axes objects

**Interactions with Use Cases**

**Use Case 1:** Computational biologist desires to assess the sensitivity of steady state to varying parameters.

1. Create an **OneWayExperiment** or **TwoWayExperiment** object using user input of model, factors, and levels. All possible experimental conditions are generated by the **set_condition**.
2. Call **calc_steady_state** to calculate steady state for every possible experimental condition.
3. To plot steady state concentration or reaction rates as a function of one variable, experiment class's **get_values** methods extracts selection values from simulations or steady state results. The values are plotted with plot_line.
4. To plot steady state concentration or reaction rates as a function of two variables, experiment class's **get_mesh** methods extracts selection values from simulations or steady state results and transforms them using **vector_to_mesh**. Conditions are converted into mesh grids as well. The values are plotted as 2D contours or 3D plots with **plot_mesh**.

**Use Case 2:** Computational biologist desires to characterize the sensitivity of time courses to varying parameters (starting concentration and/or rate constants).

1. Create an **OneWayExperiment** or **TwoWayExperiment** object using user input of model, factors, and levels. All possible experimental conditions are generated by the **set_condition**.
2. Call **simulate** to run simulation for every possible experimental condition.
3. Plot time courses over a range of values using **plot_timecourse_mesh**.
4. Cluster time series of desired species concentration/flux across conditions using **cluster_timeseries**. Visualize cluster labels for using **plot_timecourse_clusters** for 1-way experiment and **plot_mesh** for 2-way experiment.
5. (Not yet implemented) Calculate time position of first global maximum for desired species concentration/flux.

**(Not Implemented) Use Case 3:** Computational biologist desires to assess the sensitivity of oscillation to varying parameters.

1. Create an **OneWayExperiment** or **TwoWayExperiment** objects using user input of model, factors, and levels. All possible experimental conditions are generated by the **set_condition**.
2. Call **simulate** to run simulation for every possible experimental condition.
3. Plot time courses over a range of values using **plot_timecourse_mesh**.
4. Extract oscillation metrics using **fit_sinusoid** (Not yet implemented).
5. Plot oscillation metrics using **plot_line** for 1-way experiment and **plot_mesh** for 2-way experiment.

**Preliminary Plan**

By 11/19/21

- [x] ~~Experiment classes~~
- [x] ~~set_conditions~~
- [x] ~~simulate~~
- [x] ~~Calculate steady states~~
- [x] ~~get_values~~
- [x] ~~Data transformation functions~~
- [x] ~~get_mesh~~

By 11/26/21 (Showable state)

- [x] ~~plot_mesh~~
- [x] ~~plot_line~~

**Stretch Goals:**

- [x] ~~cluster_timeseries~~
- [x] ~~kmeans~~
- [ ] fit_sinusoid
- [ ] find_max
- [ ] kshape
- [ ] dbscan
- [ ] Data structure conversions to AnnData for read and write