**User Guide**

**Input file**

The input file is loosely based on the bngl format. There are specific blocks for definining parameters, species and pairwaise binding rules.

**Parameters:**

The parameter block is flagged by the “begin parameters” and “end” keywords. The parameter block is used to define certain kinetic protocols or set values of association rates or can also be used to define variables. The current list of parameters are:

**default\_assoc**: Sets the association rates of all reactions to this value.

Type: float ; Default value : 1.0 ; Units : uM-1 s-1

**creation\_rate:** Sets the default rate of titration of all creation reactions.

Type: float ; Default value : 1.0 ; Units : uM s-1

**destruction\_rate:** Sets the default rate of desctruction of all 1st order destruction.

Type: float ; Default value : 1.0 . Units : s-1

**max\_subunits:** This value decides the size of the final assembled complex. Only applies to a homo- oligomer assembly (also acts as a flag).

Type: int ; Default Value: -1.0.

**max\_interactions:** To specify the max number of interactions (binding interfaces) a subunit can have in a homo-oligomer assembly. This parameter is only considered for homo-oligomeric assemblies.

Type: int ; Default Value: 2.0 (Only useful if max\_subunits > 0)

**titration\_time\_int:** Sets the target conc of all titration reactions. Titration is stopped after system reaches the target conc. Required for creation reactions. Type: float, Default Value: -1.0 ; Units : uM.

**rxn\_coupling:** Set to true to enable the Diversification protocol (as described in the paper)

Type : bool ; Default Value: False

**homo\_rates:** Set to true to enable the Rate Growth protocol (as described in the paper)

Type: bool ; Default Value: False

**monomer\_add\_only:** Set to true to enable only monomer growth of assembly (does not allow intermediates to combine). Type: bool ; Default Value: False

**chaperone:** Set to true to enable Enzymatic recycling/dissociation protocol (see paper)

Type: bool : Default Value: False

**Species:**

Set the initial concentrations of each species (Units: uM). The species block is flagged by the “begin species” and “end” keywords. Only monomer species can be set. All other species are which are formed through the pairwise binding rules have a default starting conc. of 0.

**Observables:**

Flagged by keywords “begin observables” and “end”. Species to track throughout the simulation. Currently the code ignores this section. To get conc. profile of all species, we need to directly access the simulation data in the VecSim class.

[TO DO: Make this process user friendly]

**Rules:**

Flagged by keywords “begin rules” and “end”. This section is used to define possible allowed pairwise interactions and the stability of each interaction. The reaction network is built using these binding rules as it creates all possible species which contain combinations these interactions. Each species exists in only one topological form. Example to define the assembly of a tetrameric complex ABCD, the binding rules are as follows:

A(a)+B(b)<->A(a!1).B(a!1) G=-20

A(b)+M(a)<->A(b!1).M(a!1) G=-20

A(c)+S(a)<->A(c!1).S(a!1) G=-20

B(b)+M(b)<->B(b!1).M(b!1) G=-20

M(c)+S(b)<->M(c!1).S(b!1) G=-20

B(c)+S(b)<->M(c!1).S(b!1) G=-20

This is an example of a fully connected tetramer complex specified by writing all possible pariwise interactions between it’s monomers (4 choose 2 = 6 interactions). Each line has a reaction block and a ΔG value seperated by a whitespace. Whitespaces are **NOT** tolerated within the reaction block. Reactants and products must be separated by a “<->” character. The strength of each interaction is specified in units of *k*BT.

Creation and destruction reactions are specified by using the “null” keyword. Example:

null<->A(a) G=0 ; Creation reaction. Value of ΔG is irrelevant.

Currently the reaction network does not process interfaces on species. It is just there for aesthetic purposes.

**Running a Kinetic Simulation**

With a defined set of species and binding rules, we can run a kinetic simulation to obtain a concentration profile vs time for different species.

To do so, we can use the template Jupyter notebook in the test\_runs folder. We first start with defining a reaction network for our desired system of interacting species.

**Optimization of reaction rates**

Reaction rates can be optimized w.r.t to a desired objective function. Example files for different protocols of optimization are given in the results folder.

**Classes:**

**ReactionNetwork:**

This is the class the reads the input files and creates a Networkx structure with species represented as nodes and edges hold the reaction data. All other intermdeiate and final assembly complex species are created from the binding rules specified in the input file. For a homo-oligomer assembly additional parameters have to be defined in the parameter block.

**VectorizedRxnNet:**

This represents all relevant data like reaction rates, concentrations, free energies and other simulation parameters in vectorized form (Tensors). Different kinetic protocols like RateGrowth, Diversification etc.. require different initializations of Tensors that hold the reaction rates. This class also includes different methods for vectorized operations that are evaluated during a simulation.

**VecSim:**

This class holds all data required to run a deterministic simulation. The simulation is performed by the simulate() method. Again each kinetic protocol with different constraints require different mode of calculation of the reaction rates. At the end of the simulation, this method returns the final yield of a species (as specified by the yield\_species argument) and other outputs that vary for different kinetic models. This class also includes a method to plot concentration data vs time (plot\_observables)

**Optimizer:**

This class is the main class hold the relevant data and method to perform the optimization. Currently two modes of optimization can be perfomed – “Adam” and “RMSprop”. Additionally each kinetic model requires different instances of parameter definitions for each optimizer instance. Parameters related to the learning rate are also included in the object call. The optimization code is present in the optimize() method. Each kinetic protocol requires different forms of the loss function.

**Eq\_Solver:**

Sympy based numerical solver of a set of non-linear system of equations. This class can be used to determine the equilibrium concentration for a system under a particular condition.