Tools for Stochastic Modelling of Gene Networks

Sebastián Jaramillo-Riveri,
AIV Master Program, Université Paris Decartes, France
sebajarar@gmail.com

Supervisor: Yaakov Benenson,
Department of Biosystems Science and Engineering, ETH, Switzerland

July 4, 2013

Abstract

For Synthetic Biology to become a standard engineering discipline, predictive strategies for the design and optimisation of synthetic genetic circuits are critical[1]. Due to the complexity of biological systems and the many possible combinations, theoretical modelling is needed. Particular is the case of mammalian systems known for their stochasticity[6, 2]. Furthermore, many laboratories rely on transient transfection, which makes gene copy numbers per cell highly variable.

We have explored current methods for stochastic modelling of gene circuits, and implemented a pipeline for simulation of reaction models in MATLAB and C, compatible with SimBiology Toolbox. The implementation is based on the "First Reaction Method", chosen for its generality. To account for species with variable initial condition (such as gene copy numbers), the pipeline supports initial states sampled from marginal distributions constrained by correlations. The user can specify different kinds of simulations, parameter scans, and plotting options.

We hope that these tools would be helpful in the design and optimisation of future synthetic circuits. Further work would include model validation, addition of alternative simulation approaches, parameter inference, improvement of user interface, and increase the efficiency of the implementation.

Contents

1 Theoretical Background

We should briefly introduced the theoretical background related to the simulation of reaction based models. Many of these notes are based on the course "Computational System Biology, Stochastic Approaches" by M. Khammash.

1.1 Deterministic Reaction Models

In general, in the deterministic setting we deal with Ordinary Differential Equations (ODEs) of the form

$$\begin{aligned} \tfrac{dX_1}{dt} &= f_1(X_1, \ldots, X_n) \\ &\cdots \\ \tfrac{dX_n}{dt} &= f_n(X_1, \ldots, X_n) \end{aligned}$$

where X_i represent the concentration of a single chemical specie. The functions f_i is the derivative of the concentration, which is a function of the system's state. Each reaction will contribute to the derivative

$$\frac{dX_i}{dt} = \sum_{j=1}^{\mathsf{M}} S_{ij} r_j(X_1, \ldots, X_n)$$

where S_{ij} is the stoichiometry of specie X_i in reaction j, and $r_j(X)$ is the rate of reaction j as a function of the species concentration.

We can classify chemical reactions into two kinds: *elementary* and *complex*. Elementary reactions are thought to be the result of a single collision event, for example $A \to B$, or $A+B \to C$. The rate of these reactions can be formulated as

$$r_j = k_j \prod_i (X_i)^{S_{ij}}, S_{ij} > 0$$

meaning that is proportional to the product of all reactants (left hand side of the reaction), elevated to their stochiometric coefficient. This is the so called *mass-action law*. For example, the reaction $A + B \to C$ would have rate proportional to $[A][B]$[1].

Complex reactions are thought as a concatenation of more than one elementary reaction, thus possibly having rate laws different from mass action. The final function depends on the mechanism. Examples are *Michaelis-Menten* and *Hill* equations.

It is important to notice that reaction rate constants are not necessarily "constant". Such parameters might in turn depend on variables hidden to our model, that happen not to vary much in the context the model was developed. This ultimately depends on the granularity of the model. It is important to be aware of potential dependencies, specially if we want to extrapolate the results of our models. For example, is common to find RNA synthesis rate constants used for gene expression models. These depend on available RNA polymerase and upstream regulatory sequences. In other contexts, such as other cell lines, or different growth conditions, they may prove to have different values.

[1]We will use brackets to denote a specie concentration

1.2 Stochastic Interpretation

Conceptually, reactions are not continuous processes: each reaction have a discrete number of events per time. And more, each event modifies the system in discrete fashion. The stochastic interpretation of a reaction model attemps to correct for these facts.

Instead of using concentrations, we will use species copy numbers as our state space, and model reactions as stochastic events that modify the system's state. We should assume *well stirred* conditions, meaning we will avoid any kind of spatial inhomogeneities and diffusion processes.

Let's call a vector with copy number of every specie in our model $X(t) \in \Omega$. $X(t)$ is a random variable. We will be interested in the probability of finding the system in state x at time t given some initial state x_0 and time t_0:

$$\mathsf{P}(\mathsf{x}, t | \mathsf{x}_0, t_0) = \mathbb{P}(X(t) = \mathsf{x} | X(t_0) = \mathsf{x}_0)$$

Each reaction (from a total of M) is a possible event. We will assume that the number of events in time is a *Poisson Process*. The rate of the Poisson Process, meaning the average number of events occurrences per unit of time, is called *propensity*, and is a function of the system's state ($a(\mathsf{x})$). Also, we will call ν a vector specifying how a single event modify the state space.

Following the laws of probability we find that $\mathsf{P}(\mathsf{x}, t)$ is governed by the following equation

$$\frac{d\mathsf{P}}{dt}(\mathsf{x}, t) = \sum_{j=1}^{\mathsf{M}} \left[a_j(\mathsf{x} - \nu_j)\mathsf{P}(\mathsf{x} - \nu_i, t) - a_j(\mathsf{x})\mathsf{P}(\mathsf{x}, t) \right]$$

This is the so called Chemical Master Equation (CME). For convenience, we have defined $\mathsf{P}(x, t) = 0$, for all $x \notin \Omega$. Notice that this equation gives an ODE that describes the evolution in time of the *p.m.f* for each possible state of the system, instead of one ODE per specie as in the deterministic approach.

1.3 Meso and Macro Scales

In principle, if we were to divide the species states by the systems volume V, we would recover species concentrations. In order to ensure the continuity assumption of the deterministic approach, we should take the limit of $V \to \infty$, keeping the ratio $\frac{\mathsf{x}}{V}$ constant. This is the so called thermodynamic limit.

We will not cover how derive continuous limit [5]. However, we should mention that there is a known equivalency between the kinetic constants from the deterministic rate (Macro scale) to the stochastic propensity (Meso scale) [4].

Reaction	Deterministic Rate	Stochastic Propensity	Equivalence
$S_1 \longrightarrow S_2$	$k * [S_1]$	$c * x_1$	$c = k$
$S_1 + S_2 \longrightarrow S_3$	$k * [S_1][S_2]$	$c * x_1 * x_2$	$c = \frac{k}{V}$
$2 S_1 \longrightarrow S_3$	$k * [S_1]^2$	$\frac{c}{2} * x_1 * (x_1 - 1)$	$c = \frac{k}{V}$

where x_1 represents the copy number of S_i, and V is the systems volume.

1.4 SSA Algorithm

In many occasions finding analytical solutions to the CME it is unfeasible, or very challenging. In those cases, we have to rely on approximations.

Numerical algorithms that rely on simulation to compute probability distribution are referred as *Monte Carlo Methods*. For chemical systems, one of the simplest is the so called called Gillespie's "Stochastic Simulation Algorithm" (SSA), or "First Reaction Method" [4].

The main idea is that given a state of the system, we can compute the distribution for the time it takes for some event to happen (Δt), and the distribution that the next event is a given reaction. Then we sample from those distributions, and update the time and state accordingly.

Say that a system has a single possible reaction (r_j). As we are dealing with Poisson Processes, the time between events (Δt) follows an exponential distribution

$$\Delta t \sim a_j(\mathsf{x}(t))e^{a_j(\mathsf{x}(t))}$$

where a_j is the propensity function of the reaction. In case we have many reactions, we can compute the minimum of all inter-arrival times (equivalent to ask the time for some next event to happen). It is known that the minimum of exponential distributed random variables follows also an exponential distribution

$$\Delta t \sim a_0(\mathsf{x}(t))e^{a_0(\mathsf{x}(t))}$$

where $a_0 = \sum_j a_j$, and the probability that the minimum corresponds to reaction r_j is $\frac{a_j}{a_0}$. Then to construct the algorithm we need is to sample Δt and r_j appropriately. This is illustrated in the following iterative procedure 1.

Algorithm 1 First Reaction Method

Require: Initial condition: $T(0)$,$X(0)$. t_{max} is the upper time bound for simulation. For each reaction indexed by $j \in [1, M]$, a_j is its propensity function, and ν_j its update state vector. Finally, **rand** is a random sampler from the uniform distribution.

```
1:  $t \leftarrow T(0)$                                      ▷ Initial time
2:  $x \leftarrow X(0)$                                      ▷ Initial state
3:  $i \leftarrow 0$                                         ▷ Event counter
4:  while $t < t_{max}$ do                                   ▷ Simulate until a given time
5:      for $j \leftarrow 1$ to $M$ do                       ▷ Calculate reactions propensities
6:          $\lambda(j) \leftarrow \mathbf{a_j}(x)$
7:      end for
8:      $\lambda_0 \leftarrow \sum_j \lambda(j)$             ▷ Propensities sum
9:      $r_1 \leftarrow \mathbf{rand}([0,1])$
10:     $\Delta t \leftarrow \frac{\log(\frac{1}{r_1})}{\lambda_0}$   ▷ Sample time increment
11:     $t \leftarrow t + \Delta t$                          ▷ Update time
12:     $r_2 \leftarrow \lambda_0 * \mathbf{rand}([0,1])$
13:     $csum \leftarrow 0$
14:     for $j \leftarrow 1$ to $M$ do                       ▷ Sample next reaction
15:         $csum \leftarrow csum + \lambda(j)$              ▷ Cumulative sum of propensities
16:         if $csum \geq r_2$ then                          ▷ $j$ is the next reaction
17:             $x \leftarrow x + \nu_\mathbf{j}$            ▷ Update current state
18:         end if
19:     end for
20:     $i \leftarrow i + 1$                                 ▷ Update number of events
21:     $X(i) \leftarrow x$                                  ▷ Save state sequence
22:     $T(i) \leftarrow t$                                  ▷ Save time sequence
23: end while
```

Each simulation, is a sample trajectory of the system. By taking many independent instances, we can compute an approximation for the distributions, or values of interest. Our implementation in C of this algorithm is slightly different, as we only update the propensities after a reaction has modified the state on which it depends on, instead of doing it in every iteration.

Given that we are simulating every single reaction event and the time increment is dependent on the sum of propensities these simulations may take a long time. The higher the propensities, the higher is the *computation time* needed to simulate a given increment in *physical time*. As propensities depend on the state of the system (often linearly), when some specie gets very high copy numbers the *computation time* increases.

1.5 Random Initial Conditions and Correlations

In the previous section, we have talked of the probability of a given state in time, given some initial state $\mathsf{x}(0)$. That state could be fixed, or be a random variable. Possible examples of random initial states are gene copy numbers after transfection, or a given molecule copy number after cell division. In the case initial condition are random variables, the previous SSA algorithm remain valid; however we need to make sure to sample carefully $\mathsf{x}(0)$.

Instead of dealing with the joint probability for each possible initial state, we will work with marginal mass density functions,*i.e.* the mass density of each specie independently of any other, and pair-wise correlations. We will call π the marginal m.d.f of all species, and x' a random variable with distribution π.

We will show how to approximate x' into x by using correlations. First, we will introduce some notation. The covariance of two random variables X and Y is defined as

$$cov(X, Y) = \mathbb{E}[XY] - \mathbb{E}[X]\mathbb{E}[Y]$$

the correlation coefficient is defined as

$$corr(X, Y) = \frac{cov(X, Y)}{\sqrt{var(X)var(Y)}} = \frac{cov(X, Y)}{\sigma_X \sigma_Y}$$

In matrix notation, we should define the covariance matrix Q of the initial state as

$$Q = \mathbb{E}[\mathsf{x}(0)\mathsf{x}(0)^T] - \mathbb{E}[\mathsf{x}(0)]\mathbb{E}[\mathsf{x}(0)^T]$$

where ^{T} denotes the transpose operator. Then the correlation matrix is defined as

$$C(\mathsf{x}(0)) = Q(\sigma^{-1})^T(\sigma^{-1})$$

where σ is a vector composed of the individual standard deviations of each specie initial state variable, and $^{-1}$ is the inverse of the vector.

From C and the marginal distributions π, we can compute Q. Then, we can sample x by

$$\mathsf{x} \sim \mathbb{E}(\mathsf{x}') + L\left[(\mathsf{x}' - \mathbb{E}(\mathsf{x}')).*\sigma_\pi^{-1}\right]$$

where L is the Cholesky factorisation of Q ($Q = LL^T$), and $.*$ represents component wise multiplication of vectors. This results in a random variable that is consistent with the marginal distribution π and the correlation matrix C.

2 Implementation and tools in MATLAB

We implemented a pipeline for simulation of reaction models in MATLAB and C, compatible with SimBiology Toolbox. To account for species with variable initial condition (such as gene copy numbers), the pipeline supports initial states sampled from marginal distributions constrained by correlations. The user can specify different kinds of simulations, parameter scans, and plotting options.

The top level commands for running simulations are read from a text file with .job extension. In there, we can specify the reaction model, change initial states and parameters, and specify the types of simulations. We will illustrate the use of our pipeline with examples. All files can be found in folder doc/test/.

2.1 Top Level Execution

We will begging by introducing how to import a model, and the location of output files. .job files are *attribute* based text files. The first column corresponds to the attribute, and the second to its *value* (with the exception of VARIATE which has 2 values). Anything following # will be considered a comment. For example

```
# Example for documentation test1.job
MODEL       rna1.rxnm       # model location
OUTFOLDER   outs/           # where
DOT         1               # Make a diagram of the model
```

Here MODEL specifies the path to a the definition of the reaction based model, either in .rxnm format, or coming from a Simbiology project (.sbproj extension)[2]. For example

```
MODEL       rna1.sbproj     # Symbiology project
```

Internally, the model from Simbiology will be converted into .rxnm format and saved in the same folder.

We first need to ensure that our pipeline functions are in MATLAB path. For example by

```
MATLAB>> addpath('...src/'); % path to src folder
MATLAB>> parsejob('test1.job');
```

Then to execute in MATLAB, we make use of the function parsejob.

```
MATLAB>> parsejob('test1.job');
```

2.2 Reaction Model in .rxnm Format

Before moving to the other attributes, let's take a look at the model and the .rxnm file. This is a simple model for mRNA expression. Most typically mRNA expression is represented using pseudo-chemical reactions that coarse grain the mechanism of transcription. Let's call *G* a gene and *M* the corresponding mRNA. Then we can say that

$$\mathrm{G}_0 \underset{k_{10}}{\overset{k_{01}}{\longleftrightarrow}} \mathrm{G}_1$$
$$\mathrm{G}_1 \xrightarrow{k_m} \mathrm{G}_1 + \mathrm{M}$$

Here transcription and translation are coarse-grained into single events, with a constant rate. On the other hand, RNA and proteins are expected to disappear as a result of cell division, or decay (active and/or spontaneous).

$$\mathrm{M} \xrightarrow{\gamma_m} \emptyset$$

The model, including parameter values and initial conditions can be defined as

```
P: k01  0.00012
P: k10  0.001
P: km  0.01
P: gm  0.0001

S: G0  10

R: G0 <=> G1
   k01*G0
   k10*G1
R: G1 => G1 + M
   km*G1
R: M =>
   gm*M
```

[2]The code will assume that the model of interest is internally stored by the key "m1", which is Simbiology's default location

In general each line defines something from the model, with the exception of reactions that have 2 (irreversible) and 3 (reversible) lines. Parameters definition begging with *P:* followed by spaces, then a *symbol*, spaces, and a value. Species definition begging with *S:* followed by spaces, then a *symbol*, spaces, and a value; which will be used as its initial state. By default, all parameters and initial conditions not defined this way will be given a value of 0. **Symbols from parameters and species should be unique. All units of species are interpreted as molecules, volume as one cell, and time has arbitrary units**.

Reactions are defined by a line beginning with *R:* followed by a space and the reaction equation. Directionality are => for irreversible reactions, and <=> for reversible ones. The stochiometry of the reaction will be inferred from the equation by *regular expressions* matching. Any symbol written in an equation will be assumed to be a specie. Then follows the definitions for the reaction rates: the first one for the *forward* reaction, and the second for the *reversible* reaction. Internally reversible reactions will be converted into two uni-directional reactions.

While simulating values of species and parameters will be replaced into the rate function by regular expressions. Any symbol found in the rate function not previously defined will be assumed to be a parameter.

Other definitions include distributions (*D:*) and correlations (*C:*), and they will be introduced later. For now, the syntax goes as

```
D: G0:mdf.dat
D: G1:mdf.dat

C: G0 G1 0.8
```

where mdf.dat is an attribute file defining the marginal mass density function of the variable, and 0.8 is the correlation coefficient of the correlation of the variables at the initial state.

In case you are interested in the internal structure of the model, you can run the following command in MATLAB

```
MATLAB>> model = parserxnm('rna1.rxnm');
```

2.3 Output Folder and DOT Graphs

The OUTFOLDER attribute specifies where outputs and related files will be saved. In case the folder does not exits, it will be created (default value is the current folder).

The DOT specification will attempt to make a graph with the main relationships between species, parameters, and reaction in the model (see figure 1). It requires the package *graphviz* to be installed.

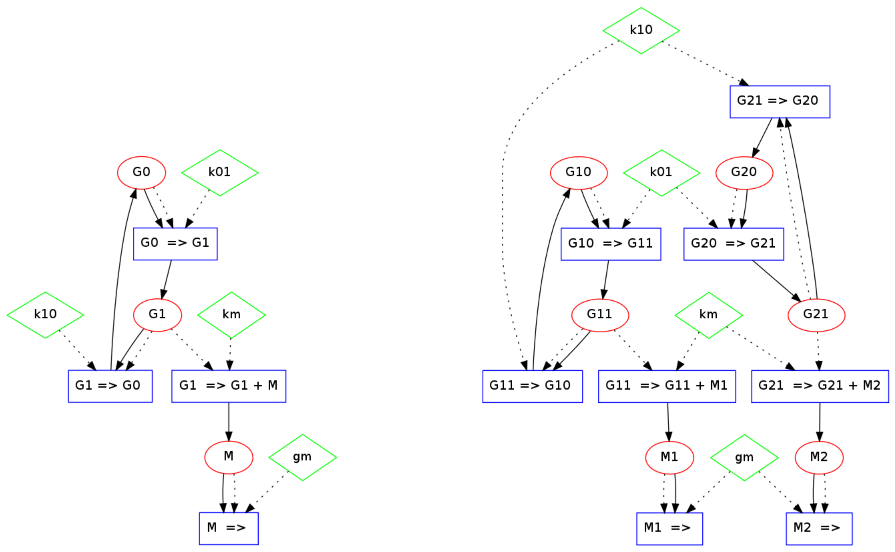

Figure 1: Graph representation of *rna1.rxnm* model (left), and *rna2.rxnm* model (right). Species are in red circles, parameters in green diamonds, and reactions in blue rectangles. Solid arrow represent stoichiometric relations, meaning when a reaction consumes or produces a specie. Dotted arrows represent rate function dependency, meaning all species and parameter that the rate function for a given reaction depends on.

Now, let's take a look at the simulation specifications. The kinds of simulations to perform are determined by TYPE attribute. DET corresponds to deterministic, and SS and ST are stochastic. SS would keep track of the state of the system at certain time points, and ST will average the state of the system over certain time intervals. Further, we have two SCAN options that will be explained later

2.4 Deterministic Simulation

We tell the pipeline we want to do a deterministic simulation, with the TYPE attribute and value DET. Time points are determined by the DETPOINTS attribute. All time points values are interpreted the same as MATLAB syntax. For example

```
# Example for documentation test1.job
NAME        doc1
OBSERVE     G0,G1,M
TYPE        DET
DETPOINTS   0:100:100000
```

The NAME attribute is a mnemonic string that will be incorporated to all the output files. The value of OBSERVE is always species symbols separated by commas. When defined, it will plot the time trajectories of those species after completing the calculations.

Also, the program will save the results from the simulation into rna1_doc1_det.mat Once completed, located in the OUTFOLDER. Output file names are constructed using the name of the model, and the NAME attribute. The variables saved are

Variable	Description
Tdet	time points
Xdet	#(time points) by #(species) matrix with the trajectories
XIDs	Cell array with the species symbols
X0	initial condition
Params	Parameters values
PIDs	Cell array with the species symbols

When OBSERVE is defined, the figure will be saved as *rna1_doc_det.fig* (see figure 2).

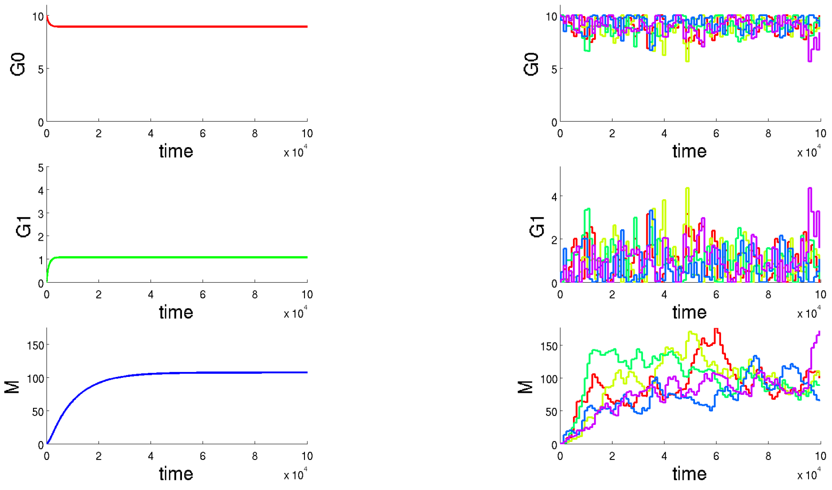

Figure 2: State of the system as a function of time in deterministic (left) and stochastic setting (right), as output from test1.job. Each colour in the stochastic trajectories correspond to a single simulation.

2.5 Stochastic and Deterministic Compatibility

We are using a single attribute to store the deterministic rate and the stochastic propensity of reactions. For this to be consistent, parameters need to be in units of molecules per cell, instead of molar concentration, and mechanisms be mass action. Rates different than mass action may bring inconsistencies. Also, automatically power functions in rates such as $k * x^2$ will be converted to $k * (1/2) * x * (x - 1)$ in the stochastic setting.

It is important to remember that all units of species are interpreted as molecules, volume as one cell, and time has arbitrary units. Meaning it is left to the user to use consistent parameter values.

2.6 Stochastic Simulations

Stochastic simulations have two main attributes the number of times the simulation will be repeated, and the times points. These are SSNREP, SSTPOINTS, and STNREP, SSTPOINTS for SS and ST types respectively. Both stochastic simulations are done in C. MATLAB will write on the fly the C code, compile it, and execute it. This code will generate output files, that MATLAB will parse and use to plot the results.

Time points have a different interpretation on each type. In SS simulations, we will keep record of the state of the system at each time point; whereas for ST simulations, time points will be interpreted as the limits of time intervals, for which the state of specie and the square of each specie will be averaged. By default, initial time will be always recorded.

2.6.1 Averaging by time intervals

The way to specify an ST simulation in the .job file is as

```
# Example for documentation test1.job
TYPE        ST
STNREP      5
STTPOINTS   0:1000:100000
```

For ST simulations, time points will be interpreted as the limits of time intervals, for which the state of specie and the square of each specie will be averaged.

MATLAB will write a C implementation of the simulation (rna1_doc1_st.c), which will save the output in text format (rna1_doc1_st...out), one file per sample.

```
0.000000 0.000000 10.000000 0.000000 0.000000 100.000000 0.000000 0.000000
0.000000 1000.000000 10.000000 0.000000 0.000000 100.000000 0.000000 0.000000
1000.000000 2000.000000 10.000000 0.000000 0.000000 100.000000 0.000000 0.000000
2000.000000 3000.000000 9.226512 0.773488 0.961511 90.061027 1.253238 3.970787
3000.000000 4000.000000 9.946531 0.053469 4.000000 133.315626 0.053469 21.493047
4000.000000 5000.000000 10.000000 0.000000 3.343315 395.868282 0.000000 38.031353
...
```

The first and second columns corresponds to the left and right limits of the time intervals. The file will always begging with 00 . . .. Then it prints the average state of each species, followed by the average of the species squared, in the same order.

Output will be saved in .mat format (rna1_doc1_st.mat) as follow

Variable	Description
Tst	time points interval limits
Xst	Cell array. One matrix for each specie of size #(time intervals) by #(samples), with the average state of the specie.
X3st	3 dimensional matrix with the average state of the specie. Size #(time intervals) by #(species) by #(samples).
STDst	Cell array. One matrix for each specie of size #(time intervals) by #(samples), with the standard deviation of the state.
STD3st	3 dimensional matrix with the standard deviation of the state. Size #(time intervals) by #(species) by #(samples).
XIDs	Cell array with the species symbols
X0	Initial states
Params	Parameters values
PIDs	Cell array with the species symbols

Average trajectory of the species from OBSERVABLE will be plotted, and saved as rna1_doc1_st.fig (see figure 2).

2.6.2 Estimating States Distributions

In SS simulations, we will keep record of the state of the system at each time point. The way to specify this in the .job file is

```
# Example for documentation test1.job
TYPE        SS
SSNREP      2000
SSTPOINTS   1000,10000,100000,500000
```

MATLAB will write a C implementation of the simulation (rna1_doc1_ss.c), which will save the output in text format (rna1_doc1_ss.out).

```
1000.000000 7 3 6
10000.000000 10 0 88
100000.000000 9 1 56
500000.000000 9 1 91
1000.000000 9 1 5
10000.000000 8 2 31
...
```

The first column are time points, followed by one for each specie. Notice that the time points are repeated, as we specified more than one repetition of the simulation.

Output will be saved in .mat format (rna1_doc_ss.mat) as follow

Variable	Description
Tss	time points
Xss	Cell array. One matrix for each specie of size #(time intervals) by #(samples), with the state of the specie.
X3ss	3 dimensional matrix with the state of the specie. Size #(time intervals) by #(species) by #(samples).
XIDs	Cell array with the species symbols
X0	Initial states
Params	Parameters values
PIDs	Cell array with the species symbols

The cumulative frequency of the species states from OBSERVABLE will be plotted, and saved as rna1_doc1_ss.fig (see figure 3).

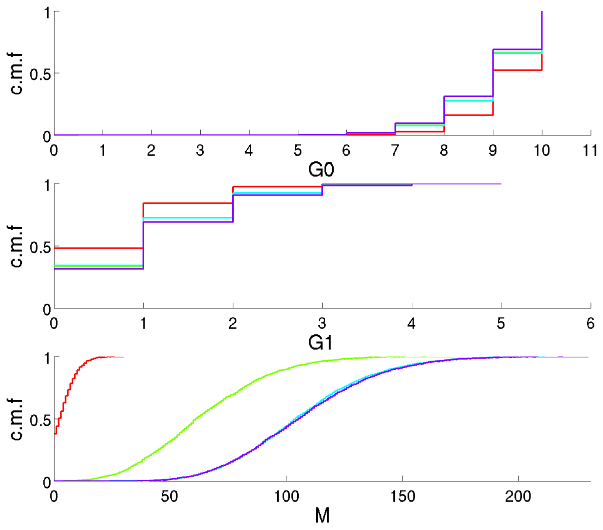

Figure 3: Cumulative frequency of species states at different time points. Each colour corresponds to a single time point.

2.7 Modifying values from the job file

It is possible to change the value of parameters and initial states inside the *.job* file. For example we can set initial state to 20, set parameter k_m to 0.1.

```
SETINIT    G0:20
SETINIT    km:0.1
```

As a matter of fact, you can add as many IDs as you want separated by commas before the semicolon, and all of those parameters or species will be set to that value.

Another feature, is when you have done the simulations, but want to observe other species. You can prevent from rerunning stochastic simulations, by changing the figures. For example, if we later become interested only in the specie M, we add to the *.job* file

```
OBSERVE    M
NOTSIM     1
```

2.8 Scatter Plots

Let's look at a variation of our previous model. We have now two genes expressing mRNA independently with the very same parameters (see figure 1).

```
# Example for documentation test2.job
NAME       doc2
OUTFOLDER  outs2/
MODEL      rna2.rxnm
TYPE       SS
SSNREP     2000
SSTPOINTS  100000
SCATTER    G10,G20
```

Here we have introduced another attribute: SCATTER. The value of this attribute is a couple or triplet of species IDs separated by commas. The result, is that a scatter plot with the results from SS type of simulation will be generated for each time point in SSTPOINTS. The points will be coloured by the density of values near by that point (see figure 4).

The figure will be saved by appending the names of the variables at the end rna2_doc2_ssG10_vs_G20.fig.

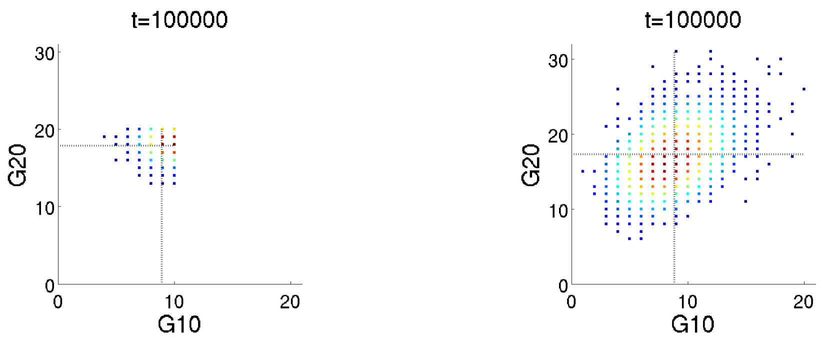

Figure 4: Scatter plots of inactive gene states ($G10$ and $G20$). Left case corresponds to a single initial state: 10 and 20 respectively. On the right, initial state of $G10$ is sampled from a Poisson distribution with mean 10, and $G20$ with mean 20. Both initial states are correlated with a coefficient of 0.6. Dotted lines represent the mean value of each variable.

2.9 Random initial States and Correlations

We will work with the model where two genes expressing are mRNA independently with the very same parameters (see figure 1).

```
# First example for documentation test3.job
NAME       doc3
OUTFOLDER  outs3/
```

```
MODEL       rna2.rxnm
TYPE        SS
SSNREP      2000
SSTPOINTS  100000
SCATTER     M1,M2
```

Now, say that the initial number of copies of each gene is a random variable. You can specify that in the .job file.

```
SETDIST     G10:poiss10.dat
SETDIST     G20:poiss20.dat
```

where poiss10.dat is a text file with two columns, the first with the values of the variable and second with the *mass density function*. In this case, it represents a Poisson distribution with parameter 10.

```
0.000000 0.000045
1.000000 0.000454
2.000000 0.002270
...
8.000000 0.112599
9.000000 0.125110
10.000000 0.125110
11.000000 0.113736
12.000000 0.094780
...
```

This will imply that the initial state of $G10$ will be sampled from a Poisson distribution of mean 10, and $G20$ from a Poisson distribution of mean 20. Both of these sampling will be independent from one another. The attribute SETDIST is not limited to one specie per time. If many species share the same distribution we could have added as many species IDs separated by commas before the semicolon, and all of those species or species will be sampled from the same distribution.

```
SETDIST     G10,G20:poiss10.dat
```

If you have a matrix (M) in MATLAB structured this way, you can save it into a file as follow

```
MATLAB>> savematrix('file.dat',M)
```

To add dependency between the initial states, we specify correlations.

```
SETCORR     G10,G20:0.6
```

Analogously to SETDIST, we could add many species before the semicolon and the correlation between all of the them will be set to 0.6.

Example scatter plots comparing the variable and constant initial states, for the gene inactive states and mRNA are in figures 4 and 5 respectively.

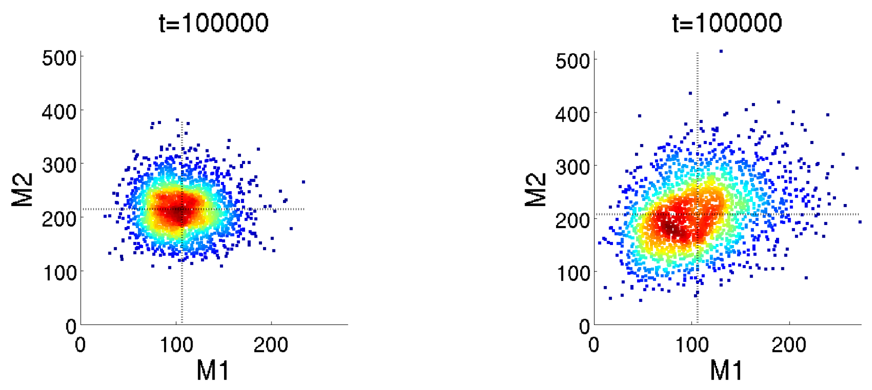

Figure 5: Scatter plots of mRNA expressed from two genes. Initial gene copies are deterministic in the right, and random variables in the left, as explained in figure 4. Dotted lines represent the mean value of each variable.

2.10 Output Scans

We have implemented shortcuts for exploring the state of the system under different parameter values and/or initial conditions. There are under the TYPE values SCANDET and SCANSS. First, we specify the model and output folder as usual

```
# Example for documentation test4.job
MODEL       rna2.rxnm # Reaction model
NAME        doc4      # Mnemonic Name
OUTFOLDER   outs4     # Output folder
```

Then, for scanning the values of the system at different gene copy numbers at time 100000; we use

```
# Example for documentation test4.job
OBSERVE     M1,M2      # Species for plotting
VARIATE     G10     10:30
VARIATE     G20     10:30
TYPE        SCANDET    # Set scan for deterministic
SCANTIME    100000     # Save the state at that time
```

SCANDET implies that it will use deterministic simulations to estimate the states, by simulating the system at every combination of values in VARIATE. VARIATE attribute is defined by two values: a symbol and a vector. The symbol could be a parameter or specie, where in the later will be interpreted as variating the initial state of that specie.

The output from the scans will be saved in .mat format (rna1_doc4_scandet.mat)

Variable	Description
variate	Symbols of parameters and species that were variated
varvals	A cell array with the values for each symbol in variate
scans	A matrix n by #(variate), where n is the number of combinations bewteen the varvals vectors. Each row corresponds to a single combination.
indexes	A matrix the same size as scans, but containing the index for each value in the corresponding vector from varvals.
scantime	Time points. By default time 0 is always included to this vector.
XscansDET	A 3 dimensional matrix #(combinations) by #(species) by #(time points), containing the state of each specie for each combination of parameters.
DefX0	Default initial states
XIDs	Cell array with the species symbols
DefParams	Default parameter values
PIDs	Cell array with the species symbols

When OBSERVE is specified, it will plot the values of those species as a function of the parameters or initial states that were variated. This is true only for scans of one or two variables (one or two VARIATE definitions). (See figure 6).

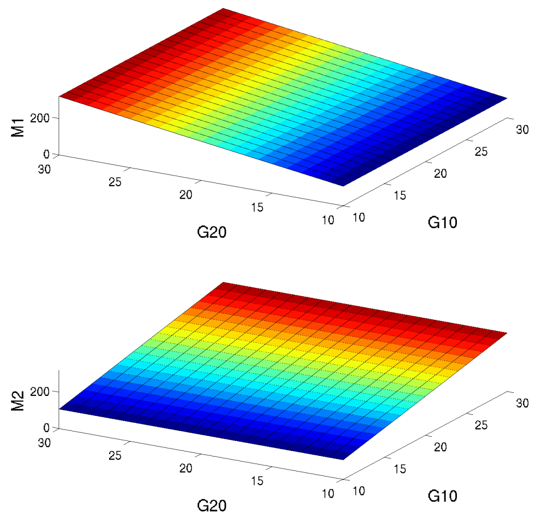

Figure 6: State of $M1$ and $M2$ at different initial conditions, at time 10^5. From deterministic simulations.

SCANSS is the analogous type, but doing stochastic simulations.

```
# Example for documentation test4.job
OBSERVE     M1,M2     # Species for plotting
VARIATE     G10      10,20,30
VARIATE     G20      10,20,30
TYPE        SCANSS    # Set scan for stochastc
SCANTIME    100000    # Save the state at that time
SCANREP     50         # Number of samples for stochastic simulations
```

SCANREP specifies how many samples are computed at each combination of the VARIATE vectors.

The output from the scans will be saved in .mat format (rna1_doc4_scanss.mat)

Variable	Description
variate	Symbols of parameters and species that were variated
varvals	A cell array with the values for each symbol in variate
scans	A matrix n by #(variate), where n is the number of combinations between the varvals vectors. Each row corresponds to a single combination.
indexes	A matrix the same size as scans, but containing the index for each value in the corresponding vector from varvals.
scantime	Time points. By default time 0 is always included to this vector.
XscansSS	A cell array, with 3 dimensional matrices #(combinations) by #(species) by #(samples) with a vector the states in all samples at final scantime
scans3	A matrix #(combinations)*#(samples) by #(variate), where each row of scan has been repeated #(samples) times.
indexes3	A matrix #(combinations)*#(samples) by #(variate), where each row of indexes has been repeated #(samples) times.
X3scansSS	A 3 dimensional Matrix #(combinations)*#(samples) by #(species) by #(time points) containing the states at each time point and condition. The rows in the first dimension correspond to the conditions in scan3.
X4scansSS	A 4 dimensional Matrix #(combinations) by #(species) by #(time points) by #(samples) containing the states at each time point and condition.
DefX0	Default initial states
XIDs	Cell array with the species symbols
DefParams	Default parameter values
PIDs	Cell array with the species symbols

When OBSERVE is specified, it will plot the average and coefficient of variation of those species as a function of the parameters or initial states that were variated. This is true only for scans of one or two variables (one or two VARIATE definitions). (See figure 7)

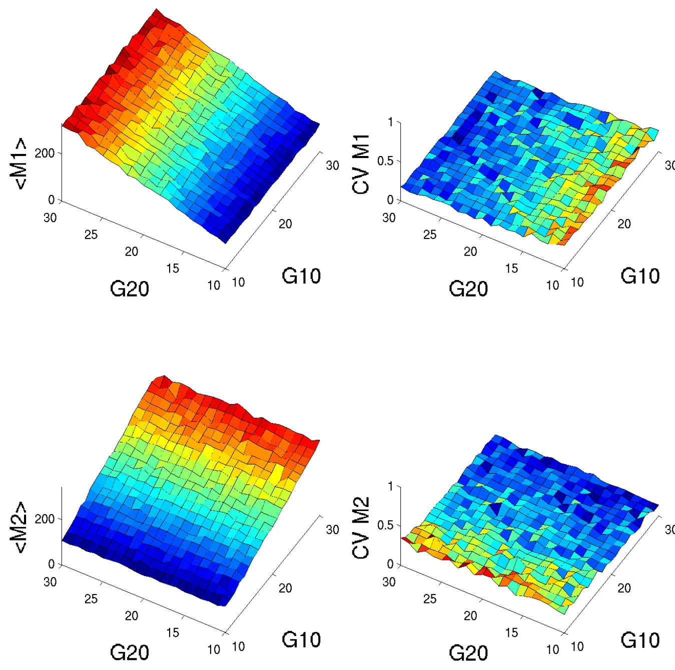

Figure 7: Average and CV of $M1$ and $M2$ copy numbers at different initial conditions, at time 10^5. From Stochastic Simulations.

3 Future Improvements

Some of the next steps that could improve these tools could be:

- Implement Gibson and Brucks stochastic method [3].
- Implement scans in C.
- Instead of writing our own C code, make it compatible with MATLAB C compiler. Then we could pass variables directly.
- Use parallel computing.
- Do unit checking before compiling.
- Implement parameters as random variables.
- Explicitly include volume and do corrections in the fly. How to handle non mass action rates is an open question to me.

References

[1] Yaakov Benenson. Biomolecular computing systems: principles, progress and potential. *Nat Rev Genet*, 13(7):455–468, 2012.

[2] Leonidas Bleris, Zhen Xie, David Glass, Asa Adadey, Eduardo Sontag, and Yaakov Benenson. Synthetic incoherent feedforward circuits show adaptation to the amount of their genetic template. *Mol Syst Biol*, 7(519):1–11, 2011.

[3] Michael A. Gibson and Jehoshua Bruck. Efficient exact stochastic simulation of chemical systems with many species and many channels. *The Journal of Physical Chemistry A*, 104(9):1876–1889, 2000.

[4] Daniel T. Gillespie. Stochastic simulation of chemical kinetics. *Annual Review of Physical Chemistry*, 58(1):35–55, 2007. PMID: 17037977.

[5] Daniel T. Gillespie. The deterministic limit of stochastic chemical kinetics. *J Phys Chem B*, 12(113):1640–1644, 2009.

[6] David M. Suter, Nacho Molina, David Gatfield, Kim Schneider, Ueli Schibler, and Felix Naef. Mammalian genes are transcribed with widely different bursting kinetics. *Science*, 332(6028):472–474, 2011.