Assignment project: CS-E5885 - Modeling Biological Networks (Spring 2019)

1 General information

Assignment name: Comparing simulation algorithms for computing dynamics of biochemical reactions.

Goal: This assignment aims at giving students a hands-on experience on strategies for simulation of biological networks. The student is required to implement and compare four different simulation strategies: 1) the Gillespie SSA, 2) the Poisson approximation method, 3) the chemical Langevin method (CLE), and 4) the deterministic simulation method. The Gillespie SSA is an exact simulation algorithm which produces an exact dynamics of the state. The Poisson approximation method and chemical Langevin method are approximate stochastic algorithms. The major difference between these algorithms is that the former treats the state as a discrete quantity, while the latter considers the state as a continuous quantity. The deterministic simulation is a highest coarse-grained algorithm which considers the dynamical behavior of the state as an deterministic and continuous quantity. The student will compare these algorithm both in terms of simulation accuracy and computational performance.

Report: Complete the above tasks and write about 3 (or more) page report briefly describing your implementations and summarizing your results, findings and other observations. Include a separate cover page containing your full names and student numbers.

Deadline: The deadline for the report is Monday, 25 Feb. (Finnish time zone). Return your report the course webpage.

Professor in charge: Harri Lähdesmäki

Academic contact person for further information on the project: Vo Hong Thanh (Email: thanh.vo@aalto.fi. Office hour: Thursday, 14:00 - 15:00, Room: B328).

2 Description of simulation algorithms

We consider a well-mixed biological network consisting of N chemical species $S_1, \ldots, S_N$ interacting through M reactions $R_1, \ldots, R_M$. The state of the system at time t is expressed by a vector $X(t) = (X_1(t), \ldots, X_N(t))$ where $X_i(t)$ denotes the discrete copy numbers (also known as *population*) of species S_i, $i = 1, \ldots, N$, at the time t. Each reaction R_j, $j = 1, \ldots, M$, denotes a possible transition of the system state. Reaction R_j has a general form

$$R_j : \; v_{1j}S_1 + ... + v_{Nj}S_N \rightarrow v'_{1j}S_1 + ... + v'_{Nj}S_N \tag{1}$$

The species involved on the left side of the arrow are called *reactants* while the ones on the right side are called *products*. The non-negative integer v_{ij} and v'_{ij}, respectively, are called *stoichiometric coefficients* which denote how many molecules of a reactant are consumed to produce a product. The amount of change in the state

caused by a firing of the reaction R_j is expressed by the state change vector v_j, a N-vector in which the ith element equal to $v'_{ij} - v_{ij}$ shows how many S_i molecules are consumed or produced by the reaction.

Suppose that the system starts at the initial state $X(0) = \mathbf{x_0}$ at time $t = 0$. The focus of this project is to predict the state $X(t)$ of the reaction network at a particular time $t > 0$. Mathematically, consider a time interval $[0, t]$. Let $N_j(t)$ be the number of firings of reaction R_j, $j = 1, \ldots, M$, in the time interval $[0, t]$. Each firing of R_j will change the state by amount v_j. The net effect of R_j to the state in the time interval $[0, t]$ is thus $N_j(t)v_j$. The dynamics of the state $X(t)$ can be written as:

$$X(t) = \mathbf{x_0} + \sum_{j=1}^{M} N_j(t)v_j \tag{2}$$

Eq.2 gives the mathematical framework for simulating biochemical reactions. The accuracy of an simulation algorithm for estimating the state $X(t)$ in Eq.2 will depend on how the algorithm calculates $N_j(t)$.

2.1 Gillespie stochastic simulation algorithm

The number of firings $N_j(t)$, $j = 1, \ldots, M$, by this simulation approach is exactly simulated. The sequence of firings of reactions will be ordered according to their firing times and the state $X(t)$ will be updated one-by-one. The derivation of $N_j(t)$ is based on the so-called *fundamental premise* of the stochastic chemical kinetics that states that the probability of a reaction R_j initiated in the next infinitesimal time interval $[t, t + dt)$ can be expressed by $a_j(X(t)dt$, where $a_j(X(t)$ is called the reaction *propensity* (also known as *intensity* or *hazard rate* in other contexts). The formula of the propensity function $a_j(X(t)$ with the mass-action kinetics is given by

$$a_j(X(t)) = c_j \prod_i \binom{X_i(t)}{v_{ij}} = c_j \prod_i \frac{X_i(t)!}{v_{ij}!(X_i(t) - v_{ij})!} \tag{3}$$

where c_j is called stochastic rate constant.

Under the premise, $N_j(t)$ denotes a Poisson process with state-dependent rate $a_j(X(t))$. The number of firings $N_j(t)$ of reaction R_j in the time interval $[0, t]$ is thus:

$$N_j(t) \sim Poi\left(\int_0^t a_j(X(t) \right), \text{with } j = 1, \ldots, M \tag{4}$$

Substitute $N_j(t)$ in Eq. (4) into Eq. (2), it gives

$$X(t) = \mathbf{x_0} + \sum_{j=1}^{M} Poi\left(\int_0^t a_j(X(t)) \right) v_j \tag{5}$$

Eq. (5) provides the rigorous framework for stochastic simulation and an algorithm that exactly simulates $X(t)$ enforced by the equation is called an exact simulation algorithm.

The Gillespie stochastic simulation algorithm (SSA) (often referred to as the direct method) is an exact simulation algorithm for sampling $X(t)$ given by Eq. (5). The algorithm makes use following facts. First, $a_j(X(t))$ changes only if state changes. Hence, before the next reaction firing, $a_j(X(t))$ remains unchanged at a value and in the following, we will denote this value as $a_j = a_j(X(t))$. Second, because $N_j(t)$, $j = 1, \ldots, M$, are independent Poisson processes with rate a_j, the next firing time of a reaction assuming that no reactions firings before is exponential distribution with rate $a_0 = \sum_{j=1}^{M} a_j$ and the reaction that fires at that time is the one with probability a_j/a_0. Algorithm 1 outlines the steps of the Gillespie algorithm.

Algorithm 1 Gillespie SSA - Direct method

Input: a biochemical reaction network of M reactions in which each reaction R_j, $j = 1, \ldots, M$, is accompanied with the state change vector v_j and the propensity function a_j

Output: a trajectory of the reaction network starting at time $t = 0$ with state $\mathbf{x}_0$ and ending at time T_{max}

initialize time $t = 0$ and state $X = \mathbf{x}_0$
compute propensity a_j for $j = 1, \ldots, M$ at state X
compute total propensity $a_0 = \sum_{j=1}^{M}$
while $(t < T_{max})$ **do**
    generate two random numbers $r_1, r_2 \sim U(0, 1)$
    compute $\tau = (1/a_0) \ln(1/r_1)$
    select minimum index μ such that $\sum_{j=1}^{\mu} a_j > r_2 a_0$
    set time $t = t + \tau$
    update state $X = X + v_\mu$
    compute propensity a_j for $j = 1, \ldots, M$ at state X
    update total propensity $a_0 = \sum_{j=1}^{M}$
end while

2.2 Poisson approximation method

This simulation approach approximately calculated $N_j(t)$, $j = 1, \ldots, M$. Here we focus on a particular algorithm called Poisson approximation algorithm. Assume that in a time interval $[t, t + \Delta t]$, the change in the propensity function $a_j(X(t))$ is negligibly small, i.e., $a_j(X(t')) \approx a_j$ for $t' \in [t, t + \Delta t]$ and $j = 1, \ldots, M$. Mathematically, let $0 < \epsilon \ll 1$ be an error control parameter. The constant propensity assumption is approximately satisfied if $|a_j(X(t')) - a_j| \leq \epsilon$ for all $j = 1, \ldots, M$ and $t' \in [t, t + \Delta t]$. The existence of the leap interval Δt is called the *leap condition*. We assume such a leap condition is existed. Otherwise, we can enforce this by repeatedly reducing the current selected time interval Δt, e.g., by a half, until the leap condition is satisfied. Under the leap condition, the number of firings $N_j(\Delta t)$ of reaction R_j in the time interval $[t, t + \Delta t]$ is

$$N_j(\Delta t) \sim \texttt{Poi}(a_j \Delta t), \text{ with } j = 1, \ldots, M \tag{6}$$

and the update of the state in the time interval $[t, t + \Delta t]$ is

$$X(t + \Delta t) = X(t) + \sum_{j=1}^{M} Poi\Big(a_j \Delta t\Big) v_j \tag{7}$$

Based on Eq. (6) and Eq. (7), the Poisson approximation method will discretize the simulation time into interval of length Δt. For each time interval $[t, t + \Delta t]$, it generates the number of firings $N_j(\Delta t)$ of R_j, $j = 1, \ldots, M$, by sampling a Poisson-distributed random number with rate $a_j \Delta t$. Then the state is updated by Eq. (7). We outline the steps of the Poisson approximation method in Algorithm 2.

Algorithm 2 Poisson approximation method

Input: a biochemical reaction network of M reactions in which each reaction R_j, $j = 1, \ldots, M$, is accompanied with the state change vector v_j and the propensity function a_j, the error control parameter $0 < \epsilon \ll 1$

Output: a trajectory of the reaction network starting at time $t = 0$ with state $\mathbf{x}_0$ and ending at time T_{max}

1: initialize time $t = 0$ and state $X = \mathbf{x}_0$
2: **while** $(t < T_{max})$ **do**
3: compute propensity a_j for $j = 1, \ldots, M$ at state X
4: choose discrete time Δt satisfying leap condition
5: generate M Poisson-distributed random number n_j with $j = 1, \ldots, M$ from $\texttt{Poi}(a_j \Delta t)$
6: update $X = X + \sum_{j=1}^{M} n_j v_j$
7: set $t = t + \Delta t$
8: **end while**

2.3 Chemical Langevin method

The chemical Langevin method (CLE) is a further approximation of the Poisson approximation method. Assume that we can find the time interval Δt that both satisfies the leap condition, i.e., propensity $a_j(X(t))$ is approximately constant over $[t, t + \Delta t)$, $j = 1 \ldots M$, and an additional condition: $a_j \Delta t \gg 1$. The Poisson distribution $\texttt{Poi}(a_j \Delta t)$ under the condition that $a_j \Delta t \gg 1$ can be approximated by a Normal distribution with the same mean and variance $a_j \Delta t$. It is

$$\texttt{Poi}(a_j \Delta t) \approx \texttt{N}(a_j \Delta t, a_j \Delta t) = a_j \Delta t + \sqrt{a_j \Delta t}\texttt{N}(0, 1) \tag{8}$$

in which $\texttt{N}(\mu, \sigma^2)$ denotes a Normal distribution with mean μ and variance σ^2. The derivation of Eq. (8) relies on the conversion of a Normal distribution $\texttt{N}(\mu, \sigma^2)$ to the standard unit Normal distribution $\texttt{N}(0, 1)$, *i.e.*, $\texttt{N}(\mu, \sigma^2) = \mu + \sigma\texttt{N}(0, 1)$.

The state update after time interval Δt, under the CLE assumption, is thus

$$\begin{aligned} X(t + \Delta t) &= X(t) + \sum_{j=1}^{M} \texttt{Poi}(a_j \Delta t) v_j \\ &= X(t) + \sum_{j=1}^{M} a_j v_j \Delta t + \sum_{j=1}^{M} \sqrt{a_j \Delta t}\texttt{N}(0, 1) v_j \end{aligned} \tag{9}$$

Eq. (9) is called the chemical Langevin equation (CLE), hence the name of the method. The equation provides the mathematical basis for the CLE method described in Algorithm 3. We note that because the state update in Eq. (9) involves the computation of the square root $\sqrt{a_j \Delta t}$, the state $X(t)$ in CLE is no longer an integer vector. This is an important difference between the Posson approximation and CLE method. The state $X(t)$ during the simulation by CLE must be represented as a vector of floating point numbers and then is converted back to integer values at the end of the simulation. We also remark that the CLE method in Algorithm 3 is equivalent to the Euler-Maruyama method in stochastic differential equation (SDE).

Algorithm 3 Chemical Langevin method

Input: a biochemical reaction network of M reactions in which each reaction R_j, $j = 1, \ldots, M$, is accompanied with the state change vector v_j and the propensity function a_j, the error control parameter $0 < \epsilon \ll 1$

Output: a trajectory of the reaction network starting at time $t = 0$ with state $\mathbf{x}_0$ and ending at time T_{max}

1: initialize time $t = 0$ and state $X = \mathbf{x}_0$
2: choose discrete time Δt satisfying leap condition and $a_j \Delta t \gg 1$
3: **while** $(t < T_{max})$ **do**
4: compute propensity a_j for $j = 1, \ldots, M$ at state X
5: generate M unit normal-distributed random number $n_j \sim \mathtt{N}(0, 1)$
6: update $X = X + \sum_{j=1}^{M} a_j \Delta t v_j + \sum_{j=1}^{M} n_j v_j \sqrt{a_j \Delta t}$
7: set $t = t + \Delta t$
8: **end while**

2.4 Deterministic simulation

The deterministic simulation is a highest coarse-grained algorithm. It has been used widely in simulation of biochemical reactions in the literature. In the following, we provide a short derivation of the deterministic simulation using the same framework we have developed so far.

Consider Eq. (9). If the last term (called the noise term) becomes negligibly small compared with the second one, i.e., $\sqrt{a_j \Delta t} \ll a_j \Delta t$, we can omit this term. The equation becomes

$$X(t + \Delta t) = X(t) + \sum_{j=1}^{M} a_j v_j \Delta t \tag{10}$$

Subtracting X(t) in both sides of Eq. (10), then taking the limit of $\Delta t \to 0$, we get a set of ordinary differential equations (ODEs) with a general form:

$$\frac{d[X(t)]}{dt} = F([X(t)]) \tag{11}$$

where now the state $[X(t)]$ denotes the molar concentration of species (not the discrete copy numbers) and F is a function of state. The molarity is measured as the mole of the substance per litre and is denoted by M. Let V be the volume of the chemical reactor and $N_A = 6.02 \times 10^{23}$ be the Avogardo's number. The conversion between the population X_i of a species S_i to its molar concentration $[X_i]$ is:

$$[X_i] = \frac{X_i}{N_A V} \tag{12}$$

We list some examples of conversions of chemical reactions into ODEs in Table 1.

We note that the deterministic rate constant k_j of a reaction R_j *is not* the stochastic rate constant c_j. In general, c_j is the probability of firing per second of the reaction (unitless), while k_j depends on the type of the reaction. Table 2 provides the formulas for calculating deterministic reaction rate constants for the reactions considered in Table 1.

We present in Algorithm 4 a simple deterministic simulation algorithm called the forward Euler method for solving ODE in Eq. (11). The student, however, is encouraged to implement advanced algorithms, e.g., Heun's method, Runge-Kutta method, which allow producing a more accurate simulation result.

The principle of the forward Euler method is to discretize the simulation time $[0, t]$ into short time intervals of length Δt. Let $[X(t)]$ be the concentration of species at time t. The concentration of species at time $t + \Delta t]$ is computed as:

$$[X(t + \Delta t)] = [X(t)] + \Delta t \cdot \mathbf{F}([X(t)]) \tag{13}$$

The steps for implementing the forward Euler method in Eq. (13) are detailed in Algorithm 4.

Table 1: Conversion of biochemical reactions to ODEs according to the law of mass action ($[\cdot]$ indicates concentrations, k_j indicates the deterministic reaction rate constant)

Reaction	Rate	ODEs
$\emptyset \xrightarrow{k_j} A$	k_j	$\frac{d[A]}{dt} = k_j$
$A \xrightarrow{k_j} B$	$k[A]$	$\frac{d[A]}{dt} = -k_j[A]$; $\frac{d[B]}{dt} = k_j[A]$
$A + B \xrightarrow{k_j} C$	$k_j[A][B]$	$\frac{d[A]}{dt} = \frac{d[B]}{dt} = -k_j[A][B]$; $\frac{d[C]}{dt} = k_j[A][B]$
$A + A \xrightarrow{k_j} B$	$k_j[A]^2$	$\frac{d[A]}{dt} = -2k_j[A]^2$; $\frac{d[B]}{dt} = k_j[A]^2$
$A + B + C \xrightarrow{k_j} D$	$k_j[A][B][C]$	$\frac{d[A]}{dt} = \frac{d[B]}{dt} = \frac{d[C]}{dt} = -k_j[A][B][C]$; $\frac{d[D]}{dt} = k_j[A][B][C]$

Table 2: Calculating deterministic reaction rate constants from the stochastic one c_j (N_A indicates the Avogardo's number, V indicate the size of the biochemical volume where reactions occur)

Reaction	Deterministic rate constant	Unit
$\emptyset \xrightarrow{c_j} A$	$k_j = c_j/(N_A V)$	$concentration \cdot time^{-1}$
$A \xrightarrow{c_j} B$	$k_j = c_j$	$time^{-1}$
$A + B \xrightarrow{c_j} C$	$k_j = c_j N_A V$	$concentration^{-1} \cdot time^{-1}$
$A + A \xrightarrow{c_j} B$	$k_j = c_j N_A V/2$	$concentration^{-1} \cdot time^{-1}$
$A + B + C \xrightarrow{c_j} D$	$k_j = c_j (N_A V)^2$	$concentration^{-2} \cdot time^{-1}$

Algorithm 4 Deterministic simulation - Forward Euler method

Input: a system of ODEs $d[X]/dt = \mathbf{F}(t, [X])$ corresponding to a biochemical reaction system, the initial state $[X_0]$ of the system with species concentrations at time 0, the simulation ending time T_{max} and the discretization stepsize Δt.

Output: a trajectory of the biochemical system expressed in terms of molecule concentrations with discretization stepsize Δt.

1: initialize time $t = 0$ and state $[X] = [X_0]$
2: **while** $(t < T_{max})$ **do**
3: update $[X] = [X] + \Delta t \cdot \mathbf{F}([X])$
4: update $t = t + \Delta t$
5: **end while**

3 Student's Tasks

The student is required to perform the following tasks:

- implement four algorithmic approaches described in Sect. 2.1 - 2.4 with a preferred language (e.g., Matlab, R, Python, Julia, C++, Java).
- run the simulation algorithms on the Dimerisation kinetics (Sect. 7.2 in [2]), the Michaelis-Menten enzyme kinetics (Sect. 7.3 in [2]), the auto-regulatory genetic network (Sect. 7.4 in [2]) and the lac operon (Sect. 7.5 in [2]). The populations of species (average +/- standard deviation for stochastic

algorithms) are plotted and compared. For stochastic approach, the populations of species should be averaged by at least 100 independent runs.

- plot and compare the runtimes (i.e., CPU times) of simulation algorithms in simulating models in the previous task. For stochastic approach, the performance of a algorithm should be averaged by at least 100 independent runs.

The student can also implement more efficient methods of the algorithms described above (see, e.g., [1]). In this case, please justify your choice of method(s). If you choose to use a method that is not yet covered in this project description, you will need to provide a more comprehensive description of the computational method in your report as well as full reference to the publication/book/webpage/other material where the method is introduced.

References

[1] Luca Marchetti, Corrado Priami, and Vo Hong Thanh. *Simulation Algorithms for Computational Systems Biology*. Springer, 2017.

[2] Darren J. Wilkinson. *Stochastic Modelling for Systems Biology*. CRC Press, 2nd edition, 2011.