Stochastic Systems: The Gillespie algorithm.

May 30, 2018

Exact numerical scheme: the Gillespie algorithm

Consider a given chemical equation, R_1. Assume S_i, with $i = 1, 2$, to label the involved reactants (called Substrates in the original paper by Gillespie):

$$S_1 + S_2 \xrightarrow{c} 2S_1$$

In words: the (individual!) molecule of type S_1 can combine with an (individual!) molecule of type S_2 to result into two molecules of type S_1.

The probability that such a reaction will take place in the forthcoming time interval dt is controlled by:

- the number of molecules of type S_1 and S_2 and the number of possible combinations that yield to an encounter between a molecule of type S_1 and another of type S_2.
- the average probability that given a pair of molecules S_1 ed S_2, the reaction R_1 takes over.

Assume n_1 e n_2 to label the number of molecules of type 1 e 2 respectively. Then, $h = n_1 n_2$ is the number of independent combinations that result in a pair S_1 - S_2.
On the other hand c measures the probability per unit of time of reacting. Hence:

$$P_1 = chdt = cn_1 n_2 dt$$

is the probability that the reaction R_1 takes over in a given time interval dt.

So far so good! What is going to happen if we have instead a system of reactions?

How are we going to sort out which reaction is going to happen first?

Let us consider first the case where two reactions are at play, namely R_1 and R_2, specified as follows:

$$\begin{aligned} S_1 + S_2 &\xrightarrow{c_1} 2S_1 \\ 2S_1 &\xrightarrow{c_2} S_3 \end{aligned}$$

Answering to two questions is mandatory at this point:

- When is the next reaction going to occur?
- Which reaction is going to happen?

Focus on the general framework.

Imagine to have k type of molecules partitioned in the following families $(n_1, n_2, n_3...)$ and assume that those molecules can react according to M distinct reaction channels, labelled with R_i con $i = 1, .., M$.

We need to calculate the quantity:

$$P(\tau, i)d\tau$$

i.e. the probability that given the system in the state $(n_1, n_2, n_3...)$ at time t:

- the next reaction occurs in the time interval from $t + \tau$ to $t + \tau + d\tau$.
- it is the reaction R_i.

The core of the algorithm is to evaluate the, presently unknown, quantity $P(\tau, i)d\tau$.

The key idea is to split such a probability into two distinct contributions, as outlined below:

- the probability $P_0(\tau)$ that, given the state $(n_1, n_2, ...)$ at time t, no reaction would eventually occur in the time interval $(t, t+\tau)$.
- the probability P_i that the reaction i occurred in the time interval $(t+\tau, t+\tau+d\tau)$.

The second quantity can be readily evaluated. We know that

$$P_i = c_i h_i d\tau$$

where h_i refers to the number of possible combinations of the chemicals as specified by reaction R_i. c_i is instead the average probability per unit of time that the molecules could react and so give birth to the prescribed products.

To evaluate $P_0(\tau)$, the probability that no reaction occurs in in $(t, t+\tau)$, imagine to partition the inspected time interval τ into K sub-intervals, each of size $\epsilon = \tau/K$. The probability that no reaction occurred in the first interval $(t, t+\epsilon)$ is:

$$\Pi_{j=1}^{M} [1 - c_j h_j \epsilon] = 1 - \sum_{j=1}^{M} c_j h_j \epsilon + O(\epsilon)$$

On the other hand this is also the probability that no reaction would occur in the next time interval $(t+\epsilon, t+2\epsilon)$. Since we have K consecutive intervals, one can write:

$$\begin{aligned} P_0(\tau) &= \left[1 - \sum_{j=1}^{M} c_j h_j \epsilon + O(\epsilon) \right]^K \\ &= \left[1 - \sum_{j=1}^{M} c_j h_j \tau/K + O(K^{-1}) \right]^K \end{aligned}$$

Perform now the limit for $K \to \infty$. We eventually obtain:

$$P_0(\tau) = \exp\left(-\sum_{j=1}^{M} c_j h_j \tau\right)$$

from which the fundamental result follows:

The sougth probability $P(\tau, i)$

$$P(\tau, i) = P_0(\tau) a_i = a_i \exp(-a_0 \tau)$$

where we have introduced the compact notation:

$$a_i = c_i h_i$$

and

$$a_0 = \sum_{j=1}^{M} c_j h_j$$

The above expression for $P(\tau, i)$ holds for $0 < \tau < \infty$ and $i = 1, .., M$.

Starting from this setting one can construct an exact algorithm that enables one to track the dynamics of a large ensemble of microscopic constituents that have to obey to an assigned set of chemical rules (or, equivalently, whose probability $P(n, t)$ has to obey to a given Master equation).

The core idea of the computational scheme (Gillespie algorithm) is to implement a Monte Carlo strategy that is able to simulate the stochastic process represented by $P(\tau, i)$. In the following we discuss the sequential steps that we are going to consider.

- STEP 0. At time $t = 0$ assign the initial values to the variables $n_1, n_2, ...$ and to the parameters c_i. Calculate the quantities $h_i c_i$ which in practice determine $P(\tau, i)$. One can also define the time of observation $t_1 < t_2 < ...$ and the stopping time t_s.
- STEP 1. Make use of a dedicated Monte Carlo technique to generate a random pair (τ, i), which obeys to the joint probability density function $P(\tau, i)$.
- STEP 2. Make use of the values as generated above to advance the system in time by a quantity τ, while adjusting the values of the population sizes n_i implicated in the selected reaction i. After this operation is being taken to completion, calculate again the quantities $h_i c_i$ for those reactions that have experienced a change in the chemicals amount.
- STEP 3. If time t is less than t_s or if there are no reactants left into the system $(h_i = 0)$ stop the simulations. Otherwise, start again from STEP 1.

Clearly the crucial step is:

- STEP 2. Make use of a dedicated Monte Carlo technique to generate a random pair (τ, i), which obeys to the joint probability density function $P(\tau, i)$.

to which the following slides are entirely devoted.

We should generate the pair (τ, i) in accordance with the distribution $P(\tau, i)$, as calculated below. We shall illustrate the so called direct method.

To this end we shall make use of our ability to generate random numbers r obeying to a uniform distribution. Notice that τ is a continuous variable, while i is discrete.

First let us write:

$$P(\tau, i) = P_1(\tau) P_2(i|\tau)$$

The probability $P_1(\tau)$ follows from:

$$P_1(\tau) = \sum_{i=1}^{M} P(\tau, i)$$

Hence, inserting in the preceding relation:

$$P_2(i|\tau) = P(\tau, i) / \sum_{i=1}^{M} P(\tau, i)$$

Recalling the above expression for $P(\tau, i)$ yields:

$$\begin{aligned} P_1(\tau) &= a_0 \exp(-a_0 \tau) \\ P_2(i|\tau) &= a_i / a_0 \end{aligned}$$

where $0 \leq \tau < \infty$ and $i = 1, 2, ..M$.

Both probability density functions are normalized in their respective domain of definition.

$$\begin{aligned} \int_0^\infty P_1(\tau) &= \int_0^\infty a_0 \exp(-a_0\tau) = 1 \\ \sum_{j=1}^{M} P_2(i|\tau) &= \sum_{j=1}^{M} a_i/a_0 = 1 \end{aligned}$$

The idea of the direct method is to generate a random number τ in agreement with $P_1(\tau)$ and then an integer i as dictated by $P_2(i|\tau)$. The resulting pair (τ, i) will therefore obey to $P(\tau, i)$.

As we shall outline in the following, it is possible to generate a random quantity τ which obeys to $P_1(\tau)$: (i) by extracting a random number r_1 from a uniform distribution and (ii) by calculating:

$$\tau = (1/a_0)\log(1/r_1)$$

Analogously (no proof given here), one can obtain an integer random i which obeys to $P_2(i|\tau)$ by extracting a random (real) number r_2 from a uniform distribution and selecting i as the integer that fulfills the double inequivalence:

$$\sum_{j=1}^{i-1} a_j < r_2 a_0 < \sum_{j=1}^{i} a_j$$

Finally, we discuss the origin of the formula for τ. It follows from the inversion technique, a Monte Carlo method which enables one to generate random numbers from a generic pdf, by using uniformly distributed random numbers.

Assume, we wish to generate the random number x distributed as $P(x)$. By definition, $P(x')dx'$ is the probability that x falls in the interval delimited by x' and $x' + dx'$. Consider $F(x)$ defined as:

$$F(x) = \int_{-\infty}^{x} P(x')dx'$$

clearly $F(x_0)$ is the probability that x is smaller than x_0. Function $F(x)$ measures the probability for x to be smaller than x_0. $F(x)$ is the probability distribution, distinct from the probability density function $P(x)$.

The inversion method, consists in extracting a uniformly distributed random number r and then select x such that $F(x) = r$, namely:

$$x = F^{-1}(r)$$

where $F^{-1}(\cdot)$ is the inverse of the distribution function associated to the pdf $P(\cdot)$.

Calculate in fact the probability that x as generated according the the above prescriptions would fall in the interval $[x', x' + dx']$. By construction, this probability is identical to the probability that r falls in between $F(x')$ e $F(x' + dx')$. Since r is uniformly distributed, such a probability reads:

$$F(x' + dx') - F(x') = F'(x')dx' = P(x')dx'$$

Assume one needs to generate a random number distributed as the pdf:

$$P(x) = A \exp(-Ax)$$

Then $F(x) = 1 - \exp(-Ax)$ e so, by imposing $F(x) = r$ one readily obtains

$$x = (1/A) \log(1/r)$$

i.e. the formula evoked before. Notice that in the derivation we have replaced $1 - r$ with the statistically equivalent quantity r.

On the implementation: back to the birth death model

$$E \xrightarrow{b} X$$

$$X \xrightarrow{d} E$$

The initial condition:

```
time=zeros(1,tmax);
nX=zeros(1,tmax);

nL(1,1)=X;
```

The main loop:

```
for i=2:tmax,
Calculate the transition probability
a1 = b (N-X)/N;
a2 = d*L/N;
a0=a1+a2;

Gillespie recipe
r1=rand(1,1); r2=rand(1,1);
tau=-1/a0*log(r1); r2=a0*r2;
ind=1;

Update the population amount

Save the results
end
```

Recall that to obtain a random integer i which obeys to $P_2(i|\tau)$ one can extract a random (real) number r_2 from a uniform distribution and then select i as the integer that fulfills the double inequivalence:

$$\sum_{j=1}^{i-1} a_j < r_2 a_0 < \sum_{j=1}^{i} a_j \tag{1}$$

In pratice the values of a_j are summed iteratively until the obtained sum becomes larger than $r_2 a_0$. The corresponding integer j (the number of elements summed up) is the index i we are looking for.

```
while(ind),
prob=a1;
if(r2<prob),
X=X+1; ind=0; break;
end
prob=prob+a2;
if(r2<prob),
X=X-1; ind=0; break;
end
end
```