CS207 Final Project Feature Proposal: stochastic and deterministic simulation

Divyam Misra, Shiyun Qiu, Victor Zhao

November 20, 2017

1 Summary

When the abundances of chemical species participating in a reaction are low, reactant quantities no longer change deterministically. Reaction events occur stochastically. Such situations are best modeled by stochastic simulation, as opposed to deterministic, differential equation-based modeling.

For the final project for CS207, we intend to implement a stochastic reaction simulator as well as a simple ODE solver, enabling comparison between stochastic and deterministic simulation.

1.1 Stochastic simulation methodology

The stochastic simulation algorithm we will implement is the Gillespie algorithm. The principle behind the algorithm is that waiting times between reaction events are exponentially distributed; thus the time until the next reaction is drawn from an exponential distribution (the simulation is advanced in time by this amount). The particular reaction event that occurs is randomly selected from among the possible reactions in proportion to their probabilities.

The following is a summary of the simulation process:

1. Generate two random numbers r_1, r_2 uniformly distribution in (0,1).
2. Compute the propensity function of each reaction and compute
   - $\alpha_0 = \sum_{i=1}^{q} \alpha_i(t)$
3. Compute the time interval until the next chemical reaction via
   - $\tau = \frac{1}{\alpha_0} \ln[1/r_1]$
4. Compute which reaction occurs. Find j such that
   - $r_2 \geq \frac{1}{\alpha_0} \sum_{i=1}^{j-1} \alpha_i$
   - $r_2 < \frac{1}{\alpha_0} \sum_{i=1}^{j} \alpha_i$
5. The jth reaction takes place. Update the numbers of chemical species accordingly.

6. Continue simulation by returning to step 1.

To elaborate, the *propensity function* for a chemical reaction describes the probability of a particular reaction happening. For instance, consider the following bimolecular reaction:

$$A + A \to B, \qquad \text{rate}\, k \tag{1}$$

The quantity of species A present at time t is given by $A(t)$. The propensity function for the above reaction is $A(t)(A(t)-1)k$. In a more complex reaction system, each elementary reaction will have its own propensity, and it is necessary to calculate the propensity of all elementary reactions in order to find the total "propensity" for a reaction to occur, in order to draw time τ until the next reaction from an exponential distribution.

1.1.1 Further algorithmic improvements

The Gillespie algorithm described above grow more expensive and less efficient as the numbers of reactants and reactions increase. Advancements that improve the efficiency of the algorithm have been made. If sensible, such optimizations will be implemented.

1.2 Deterministic simulation methodology

To provide a basis of comparison to our simulation results, we will also implement a deterministic ODE solver. At the very least, this will be a forward Euler solver that will proceed as follows.

1. Calculate reaction rates $r(x_i(t))$ for all elementary reactions in the system (where $x_i(t)$ represents species quantities or species concentrations).

2. Update reacting species quantities via $x_i(t + \Delta t) = x_i(t) + r(x_i(t)) * \Delta t$.

3. Advance time by Δt, returning to step 1.

1.3 Proposed capabilities of our feature

Note that no provision has been made for non-elementary reactions.

We propose to enable both stochastic and deterministic simulation of reaction systems. This will enable comparison between stochastic trajectories and the deterministically predicted trajectory. Our software feature will generate trajectory data and be able to plot such data using `matplotlib`.

It would be interesting to see at what quantities of chemical species that deterministic modelling is sufficiently predictive, and also how the efficiency of our stochastic algorithm changes with the quantities of chemical species.

1.4 Plans for implementation

Currently, our chemical kinetics library returns reaction rates with a single function call (a class method). It should be simple to integrate that function call into an ODE solver class that produces the deterministic solution.

For stochastic simulation, some of our existing classes will have to be adapted to handle discrete numbers of chemical reactants. Additionally, the stochastic simulation will probably be handled by a new class, one that can store parameters, advance the simulation, and save results.

Currently, we foresee at least two new modules, one for each type of simulation.