Ace Shen

January 20, 2019

This documentation aims at providing a comprehensive, but not detailed, instructions on simulating protein aggregation problem using the Gillespie algorithm (the kinetic Monte Carlo (kMC)). The results from the simulation are then compared to the analytical solutions (approximations) using the moment closure method. For now, the program works well in the homogeneous case (the system is represented by the concentration of monomers and aggregates, no spatial dependence). Inclusion of diffusion will be the future work, although a primitive incorporation of diffusion is already in the code.

Moment closure calculation will be introduced in the first part, and the second part the description of the Monte Carlo simulations.

Questions and comments are welcome through my email: js113@rice.edu

Mathematica script for the calculations of the moment closure

The system of moment equations derived from the rate equations for protein aggregation systems is not closed. We use the derivative matching moment closure technique to close the system and then obtain numerical solutions. The calculation involving Ito calculus since we are dealing with stochastic differential equations (SDE). The Mathematica script, *moment_closure_calculation.nb*, closes the system of SDEs to the desired order, solves numerically the system of closed SDEs, and plots the population moments as a function of time. The first block in the script is the main function that takes care of the input equations, while the second block sets the initial conditions and calls the function from the first part to do the calculation. The third block is responsible for plotting. Note that you should only modify the first part if you are changing the type of aggregation reactions.

Mathematica is powerful in symbolic calculations, but it is slow regarding data processing. Thus, we use python to compare the results from the closed system of SDEs to the kMC simulation results. The *model* function in *plot_pmoments_nc2_close23.py* takes the output code from the Mathematica script, *moment_closure_generate_pythonscript.nb*, as input, and then solve it in the python environment. Note that the python script *plot_pmoments_nc2_close23.py* also reads data from the kMC simulations.

C++ code for the kMC simulations

To compile the program,
$ g++ -std=c++11 -pthread AG.cpp -o AG
works on most linux based systems. To run it,
$ ./AG parameters.ini
The program is multi-threaded, in the sense that if the input parameter $nsample > 1$, calculations in a sample will be asigned to a thread. The program will wait until all threads are done calculations and then stop.

The basic scheme of kMC is as follows. First the system would calculate the total propensity (weights for every reaction), and the randomly choose one of the reaction to occur. Then the system is updated with the result of that reaction, with a randomly generated time interval elapsed. For a homogeneous system, the program keeps track of how many l-length aggregates are in the system, and this information is written into output files in the time interval of Nt/Ndatapoint. The output files are named after the number of sample and the length of aggregate. There is another output file named *maxl.dat* comtaining the maximal length of aggregate in each sample.

The input parameters are:
$nsample$: # of samples in the MC simulation
$nstep$: # of steps for the simulation (one step one reaction happens)
$nmonomer$: total monomers in the system
xl, yl: length of the system
$xcompartment$: # of compartments along the x axis
$primnuc$: n_c for primary nucleation
$kprim$: k_n
$kadd$: k_+
$kfrag$: k_f
$koff$: k_-

$ksec$: k_2
$n2$: n_2 for secondary nucleation
$randini$: randomized initial locations of seeds
Nt: total time for the simulation (seconds)
$Ndatapoint$: # of datapoints to be recorded
$Dscale$: scaling the diffusion coeff
$Nseeds$: # of inital seeds
$.//data//sample//$: path to save the data file

The python script can read the output data from the kMC simulation and do the plotting.