CPM/Gillespie algorthim Read Me

Asheesh Momi

April 2020

The algorithm implemented is a slightly modified Gillespie Algorithm integrated with the Cellular Potts model. To understand the theory of the Gillespie algorithm read [Erban et al., 2007], this should contain enough info about the Cellular Potts Model, but if you want more info, check out [Rens and Edelstein-Keshet, 2019] (both of these pdfs are in the folder). All code was written to work with Matlab 2018b on a 2018 MacBook Pro.

1 Math model

This is the code is based of off the PDE model from [Tang et al., 2018], written here for your reference.

$$\frac{\partial \rho}{\partial t} = I_\rho(\frac{L_R^n}{L_R^n + \bar{R}^n})\rho_i - \delta_\rho \rho + D_\rho \nabla^2 \rho \tag{1}$$

$$\frac{\partial \rho_i}{\partial t} = -I_\rho(\frac{L_R^n}{L_R^n + R^n})\rho_i + \delta_\rho \rho + D_{\rho_i} \nabla^2 \rho_i \tag{2}$$

$$\frac{\partial R}{\partial t} = (I_R + I_K^*)(\frac{L_\rho^n}{L_\rho^n + \rho^n})R_i - \delta_R R + D_R \nabla^2 R \tag{3}$$

$$\frac{\partial R_i}{\partial t} = -(I_R + I_K^*)(\frac{L_\rho^n}{L_\rho^n + \rho^n})R_i + \delta_R R + D_{R_i} \nabla^2 R_i \tag{4}$$

$$\frac{\partial P}{\partial t} = B(\frac{K^n}{L_K^n + K^n})P_i - \delta_P P + D_P \nabla^2 P \tag{5}$$

$$\frac{\partial P_i}{\partial t} = -B(\frac{K^n}{L_K^n + K^n})P_i + \delta_P P + D_{P_i} \nabla^2 P_i, \tag{6}$$

where,

$$R = \frac{[RacGTP]}{[Rac_{tot}]} \quad R_i = \frac{[RacGDP]}{[Rac_{tot}]} \quad \rho = \frac{[RhoGTP]}{[Rho_{tot}]} \quad \rho_i = \frac{[RhoGDP]}{[Rho_{tot}]} \quad P = \frac{[Pax_p]}{[Pax_{tot}]} \quad P_i = \frac{[Pax]}{[Pax_{tot}]}$$

$$K = \Gamma^{-1}\alpha_r R(1 + k_X[PIX] + k_G k_X k_c[GIT][PIX][PAK_{tot}]P)$$

$$I_K^* = I_K(1 - \frac{1+\alpha_R R}{\Gamma})$$
$$\Gamma =$$
$$(1 + k_X[PIX] + k_G k_X k_c[GIT][PIX][PAK_{tot}]P)(1 + \alpha_R R) + k_G k_X[PIX][GIT]$$
$$\bar{R} = \frac{[PAK-RacGTP]+[PIX-PAK-RacGTP]+[Pax_p-GIT-PIX-PAK-RacGTP]}{[Rac_{tot}]}$$
$$\bar{P} = \frac{[Pax_p-GIT-PIX-PAK]+[Pax_p-GIT-PIX-PAK-RacGTP]}{[Pax_{tot}]}.$$

Gilespie Algorithm

1. The six main chemical species can undergo a total of 34 reactions
   - Rac inactivation $R \rightarrow R_i$
   - Rac activation $R_i \rightarrow R$
   - Rac complexing and decomplexing $R \leftrightarrow \bar{R}$
   - Rho inactivation $\rho \rightarrow \rho_i$
   - Rho activation $\rho_i \rightarrow \rho$
   - Pax inactivation $P \rightarrow P_i$
   - Pax activation $P_i \rightarrow P$
   - Pax complexing and decomplexing $P \leftrightarrow \bar{P}$
   - Moving a lattice square in the $\pm x$ and $\pm y$ directions for all six chemical species (24 reactions).

2. Reaction propensities (probability per unit time) of all possible reactions were calculated by multiplying rates from Eq. (1-6) by the number of molecules present in each cell. For example, for conversion of inactive paxillin to active paxillin, the coefficient representing gain in Eq. 5 ($B(\frac{K^n}{L_K^n+K^n})$) is multiplied by the total number of inactive paxillin molecules in each lattice square. This was done for each reaction, then iterated over every lattice square.

3. Diffusion events are modeled in a similar way. The rate of of diffusion (given by $k_{diff} = \frac{D*N^2}{L^2}$ where D is the diffusion coefficient, L is the length of cell, and N is the size of the grid), is multiplied by the number of molecules in a given lattice square. This is repeated for each of the four allowed direction of movement.

4. The reaction above were evolved according the Gillespie algorithm. Where the waiting time in between reactions τ is determined by $\tau = \frac{1}{\alpha_{tot}} ln(\frac{1}{r})$ where α_{tot} is the sum of all propensities and r is a random number drawn uniformly between 0 and 1.

5. To determine which reaction occurs we multiply α_{tot} by a separate random number r_2 drawn uniformly between 0 and 1. Then this is compared to a cumulative sum of propensities to determine which reaction occurs. For

example, if we had 4 reactions with propensities $\alpha_1 = 8, \alpha_2 = 6, \alpha_3 = 5, \alpha_4 = 1$, then $\alpha_{tot} = 20$; if $r_2 = \frac{3}{4}$, then $\alpha_{tot} r_2 = 15$. Based on this, we can conclude that $\alpha_1 < \alpha_{tot} r_2$ which means that reaction 1 will not occur, and $\alpha_1 + \alpha_2 < \alpha_{tot} r_2$ which means that reaction 2 will not occur either. However, $\alpha_1 + \alpha_2 + \alpha_3 \geq \alpha_{tot} r_2$ implying that reaction 3 will be iterated.

6. a modification to standard Gillespie algorithm is made by first determining the reaction that happens then where it happens. For example consider a 2x2 grid with 2 reactions $\alpha_1 = 8$, $\alpha_2 = 12$, and $r_2 = 0.5$. Since $\alpha_{tot} r_2 = 10$ reaction 2 happens. Now we cosider the location. Say the 4 latices have propensities for reaction 2 of (1,2,3,6). $\alpha_1 + 1 = 9 < \alpha_{tot} r_2$ but $\alpha_1 + 1 + 2 = 11 > \alpha_{tot} r_2$ so the reaction occurs in the second lattice. This formalism while more complicate is much faster and equivalent.

7. Rac and Pax are involved in complexing reactions. In [Tang et al., 2018] these where set to steady state so they are assumed to be instantaneous. This means that Rac and Pax can exist in in 3rd state complexed called $\bar{R}, \bar{P}$. In algorthim, this was implemented by executing either complexing or decomplexing reaction, determined by the probability of complexing. For active Rac the probability of complexing is given by $P_R = \frac{RK_R}{RK_R + \bar{R}}$ while for active paxillin $P_P = \frac{PK_P}{PK_P + \bar{P}}$, where

$$K_R = (1 + k_X[PIX] + k_G k_X k_c[GIT][PIX][Pax_p])\alpha\Gamma^{-1}[PAK_{tot}]$$
$$K_P = (1 + \alpha[RacGTP])k_G k_X k_c[GIT][PIX][Pax_p]\alpha\Gamma^{-1}[PAK_{tot}].$$

When any non diffusion reaction involving either Rac or paxillin occurred (as described in 2-4), the above probabilities were used to simulate the appropriate complexing reactions. This was repeated, 50 times, a sufficiently large number to represent the difference in time scales. In theory this happens infinte times we are saying $50 \approx \infty$

Cellular Potts Algorithm

In this model, space is divided into a square lattice, a fraction of which is occupied by the cell. The cell is then assigned an energy based on the Cellular Potts Hamiltonian,

$$H = \lambda_a (A - a)^2 + \lambda_p (P - p)^2 + J(i, j)P, \tag{7}$$

where a and p are predetermined values for ideal area and perimeter, A is the current area, P is the current perimeter, λ_a corresponds to the strength of pressure preventing deviations from the ideal area, λ_p corresponds to membrane elasticity, and $J(i, j)$ is a boundary term which corresponds to the cell interacting with its surroundings.

It is important to note that this Hamiltonian is phenomenological and does not actually describe the energy of the system (so it can be tweaked). This is

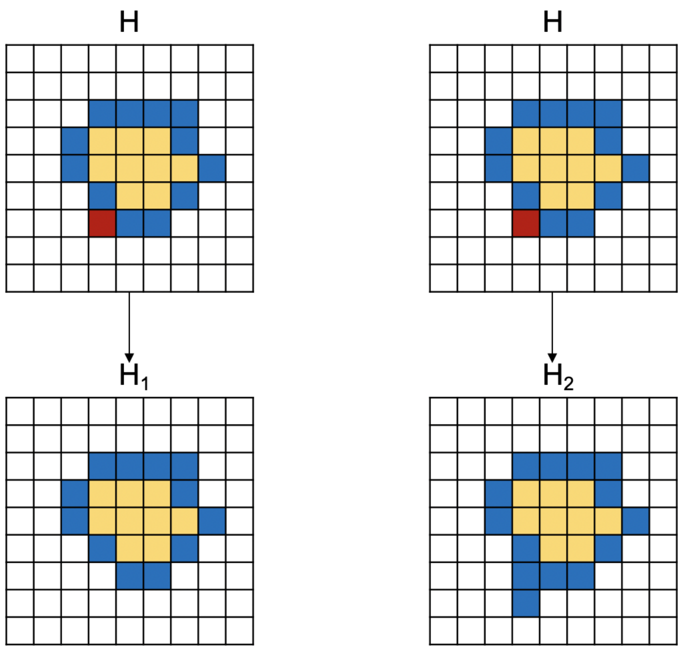

Figure 1: Cellular Potts outline. An example of two possible changes a cell can go through with the Cellular Potts Model Boundary points are shown in blue. Among these boundary points a randomly selected point is shown in red. This point can then either retract (left) or protrude (right). The changes in shape correspond to change in energy determined by Eq. 7 $\Delta H = H_1 - H$ (left), $\Delta H = H_2 - H$ (right).

then integrated with the Gillespie Algorithm in the following iterative manner.

1. To change cell shape, a random boundary point is selected and is allowed to either expand or retract, as shown in Fig. 1. The change in energy determined using Eq. 7.

2. Based on the chemical concentration of Rac and Rho in the selected square (shown in red in Fig. 1), an additional term is added to the change in energy (ΔH), to account for the Rac/Rho system. $\Delta H = \pm\beta_R(R - R_{eq}) \mp \beta_\rho(\rho - \rho_{eq})$. Where R_{eq}, ρ_{eq} are the equilibrium concentrations of active Rac and Rho respectively in a selected lattice (shown in red in Fig. 1) and β_R, β_ρ are the chemical potentials of active Rac and Rho respectively. The first term is positive in the event of a retraction and negative in the event of a protrusion.

3. The even then proceeds with a Boltzmann probability, $\{1\, if\ \Delta H \leq H_b$ $e^{-(\frac{\Delta H - H_b}{T})} if\ \Delta H > H_b,$ where T serves as a measure of stochasticity and H_b is the energy barrier for an event. Like in monte carlo simulations.

4. (1-3) are then repeated P times, where P (the perimeter) is the number of boundary points (shown in blue in Fig. 1).

5. The reaction-diffusion system is then iterated forward for a time $t = \frac{h}{v_{max}}$, where h is the length of a lattice square and v_{max} is the maximum speed of the cell you set.

Notes

- the speed is determined by the number of molecules the size of the grid and the number of lattices play around with these as to adjust speed but don't go overboard to get the code to go fast.
- You should run everything with a smaller number of molecules first to make sure there are no bugs, and everything works the way you expect.
- It is a lot faster if you place the CPM_chem_func in a C wrapper. To do that type "coder" in the command line (you may have to install it), follow the steps and create a MEX file. (I use the main to define the variables it'll try and run the entire code you can terminate early as long it runs the function once.)
- If you use a c wrapper, you'll have to redo that anytime an array changes size (it's OK this is quick.)
- the finding what cell the reaction happens in is the slow step (you can see this using a profiler).
- If you have any questions don't hesitate to email me at `asheesh.momi@yale.edu` (or ask Laurent he knows everything).

References

[Erban et al., 2007] Erban, R., Chapman, J., and Maini, P. (2007). A practical guide to stochastic simulations of reaction-diffusion processes.

[Rens and Edelstein-Keshet, 2019] Rens, E. G. and Edelstein-Keshet, L. (2019). From energy to cellular force in the cellular potts model. *bioRxiv*.

[Tang et al., 2018] Tang, K., Boudreau1, C. G., Brown, C. M., and Khadra, A. (2018). Paxillin phosphorylation at serine 273 and its effects on rac, rho and adhesion dynamics. *Computational Biology*, 14.