Will Kovacs

CS279

Gray-Scott Implementation of a Diffusion Model

Introduction

Biological systems are based on the movement and reactions of individual molecules, both within and outside of cells. In order to get a better understanding of these systems, diffusion models have been developed to provide a virtual system to quickly explore how these particles can behave in various situations. These models can range from the simple, such as a simple random walk algorithm, to more complex ones involving Monte Carlo algorithms over a 3D cellular model[1] or stochastic models that diffuse according to Brownian motion.[2] Such models have proven useful in a variety of biological scenarios; MCell, a model using Monte Carlo methods, is useful with studying calcium in the synapses of neurons[3,4], and Smoldyn has been used to simulate dimerization of a protein in a membrane[5] and to describe F-actin dynamics in T-cells[6]. Another useful model is the Gray-Scott diffusion model, which is versatile in that it can model a variety of scenarios given only slight changes in its parameters. These scenarios include the patterns on a variety of animals[7] to the growth patterns of fungi[8] to the oscillation of photoluminescence in semiconductor quantum dots.[9]

The Gray-Scott Model

This continuum model simulates two chemicals, A and B, diffusing over a grid, with A diffusing faster than B. Across all cells, A is added at a specified feed rate, while B is removed at a specified kill rate. Meanwhile, if these two share the same cell, a reaction can occur where two B particles can convert an A into a B. This can be easily modeled for each cell in the following equations:

$$A' = A + \left(D_A \nabla^2 A - AB^2 + f(1 - A)\right)\Delta t$$

$$B' = B + (D_B \nabla^2 B + AB^2 - (k + f)B)\Delta t$$

where A and B are the current concentrations of A and B in the cell, A' and B' are the new concentrations, D is their diffusion constants, ∇^2 is the Laplacian operator, f is the feed rate, k is the kill rate. In this project, I attempt to implement the standard 2D Gray-Scott diffusion model along with two variants: one where the kill rate can vary at different cells, and a 3D version.

Implementations

For the 2D implementation of the Gray Scott Model, I utilized the code from assignment 3 as a starting point, as it provides a useful grid template and visualization. I updated the model so that each block was capable of tracking the concentrations of both A and B. For the update step, I replaced the Laplacian calculation with a 3x3 convolution kernel where the current cell was given weight -1.0, adjacent were 0.2, and corners 0.05, in order to get a more accurate solution of the Laplacian than just including adjacent cells. The concentrations in the blocks were updated according to the equations above. The diffusion constants for A and B were set at typical values[10] of 1.0 and 0.5, respectively. The time step used was 0.5, with 25 steps per iteration. Initial concentration of A was set to 1 everywhere, while B had a single seed whose concentration was set to 1 at the center. The kill rate and feed rate can be adjusted before the script is run to desirable values. After testing, a grid of 50 x 50 cells was used, as it provided a wide enough space to establish a pattern, while also not requiring too much computation power per iteration.

For testing, I used two sets: a kill of .055 and feed of .062, and the other with a kill of .03 and feed of .062. These two result in rather different patterns, the former folding inwards while expanding outwards and the latter forming separate blobs, allowing for easy testing.

Visualization of the cells are based on the concentration of A, as it is guaranteed to be from 0 to 1 allowing for a static scale, and it allows to get a general sense of both concentrations of A and B as it decreases as B increases. The results of such an implementation can be seen in figure 1, along with comparison images from an online model.[11]

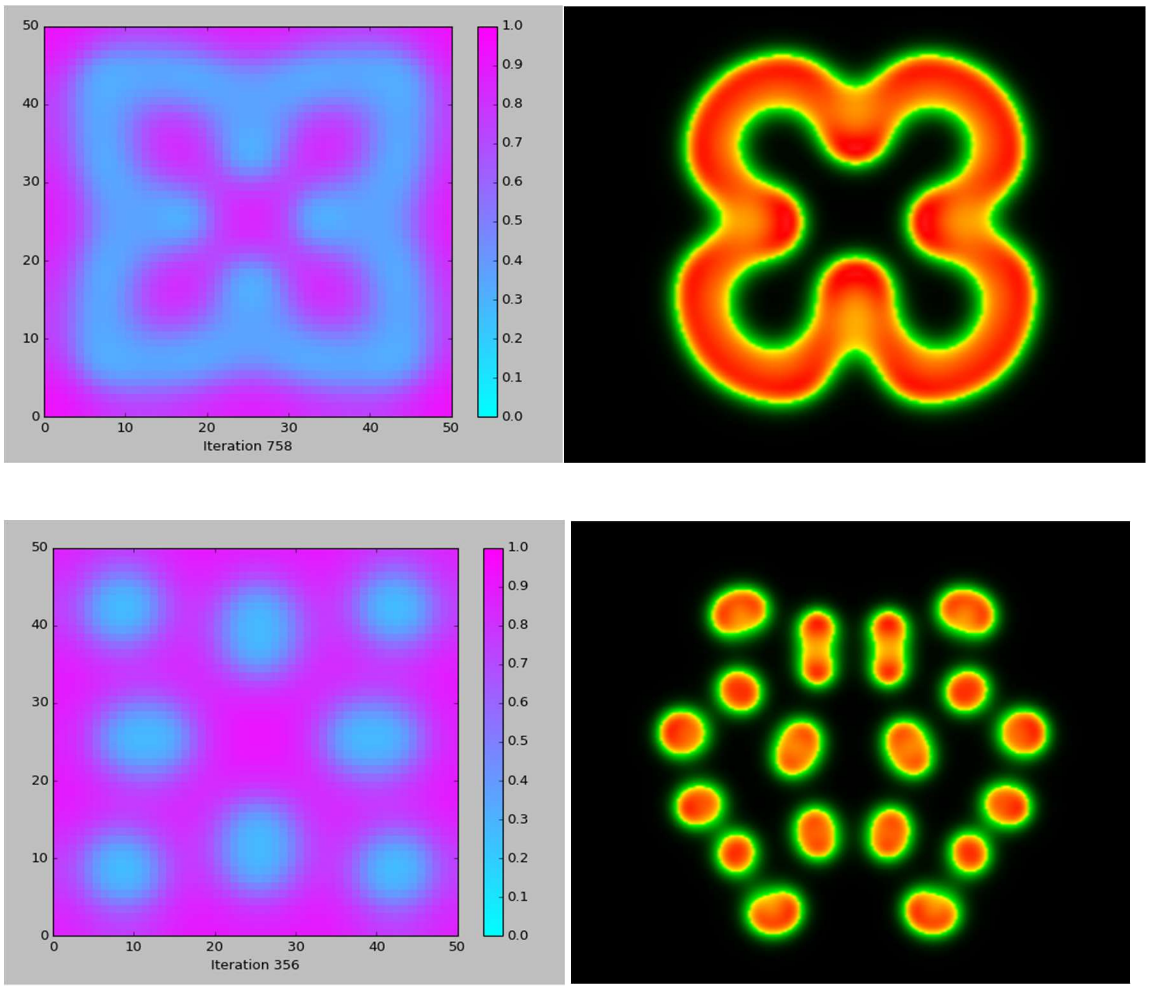

Figure 1: The top row shows the Gray-Scott model when run with a feed rate of 0.055 and a kill rate of 0.062, while the bottom row has a kill rate of 0.032 and feed rate of 0.062. The left hand side (cyan and purple) are my implementation, while the right hand side are validation images from [11].

The similarity between the two sets of images suggest a correct implementation of the Gray-Scott Model, and the differences between them are due to the images coming from different time steps, as it is impossible to synchronize them.

The next implementation allowed variation of the kill rate along different patterns across the grid. The major change to make this method work is to allow individual cells to keep track of their own kill rates. To test this implementation, simulations were run with a constant feed rate of 0.055, and alternating kill rates in concentric circles. In the NxN grid, the circles had radius N/8, 2N/8, 3N/8, and 4N/8. In one simulation, their kill rates were decreasing (0.062, 0.060, 0.058, and 0.056, respectively). The other had alternating kill rates (0.062, 0.061, 0.063, 0.060). These simulations can be seen in figure 2.

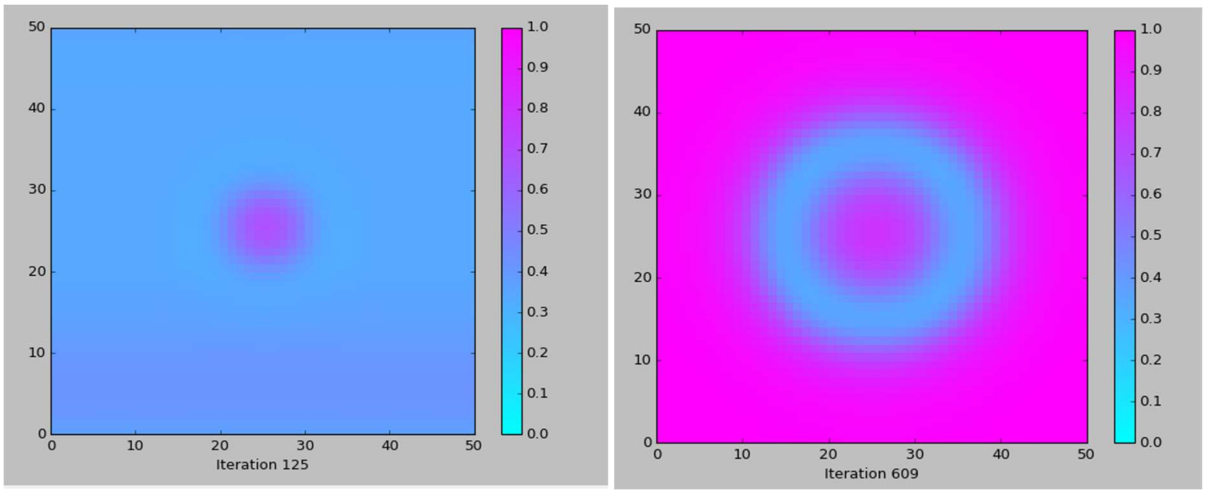

Figure 2. Simulations with changes in kill rate in concentric circles, decreasing from the center on the left side and alternating on the right.

The patterns shown make sense considering the situations of each: when the center has the highest kill rate, there will be the least concentration of B, while in the right hand side, when a ring is bounded by two rings with larger kill rates, the concentration of B will remain inside, unable to escape to the outermost. What is surprising, though, is that these changes are caused by very minute changes in the kill rates.

The final model implemented was a 3D version of the model. The major computational adjustment to add another dimension was finding a good Laplacian kernel. To this end, I decided to use a 3x3x3 kernel, whose corners had weight 0, adjacent blocks with weight 1/6 and current with weight -1. Having these weights be between 0 and 1 is important to ensure that A does not escape its bounds of 0 to 1. This extra dimension also made the overall algorithm much more computationally expensive, necessitating smaller dimensions for the cube (to be 25 x 25 x 25) in order to run in a reasonable time. Furthermore, the visualization becomes much more difficult, requiring a different method from the 2D version. After some experimenting with projections of the 2D model in the 3D space, matplotlib seemed to initially work well, as seen in figure 3.

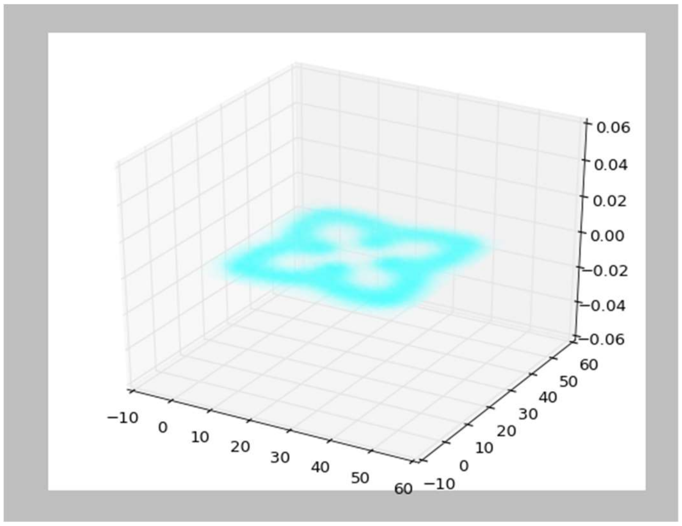

Figure 3. Projection of 2D model in 3D space using matplotlib.

For running the 3D version, a larger seed had to be included so that the diffusion doesn't die at the onset. The seed used here was the central block and its six adjacent ones. The color also had to be changed, so that it would show up better, so pink was used to represent the concentration of A. Two simulations of this method were run, similar to the 2D version: one with a kill of .055 and feed of .062, and the other with a kill of .03 and feed of .062, and images were taken when subsequent iterations produced little variation. They can be seen in figure 4, along with the XY cross section when Z = length/2 (here 13).

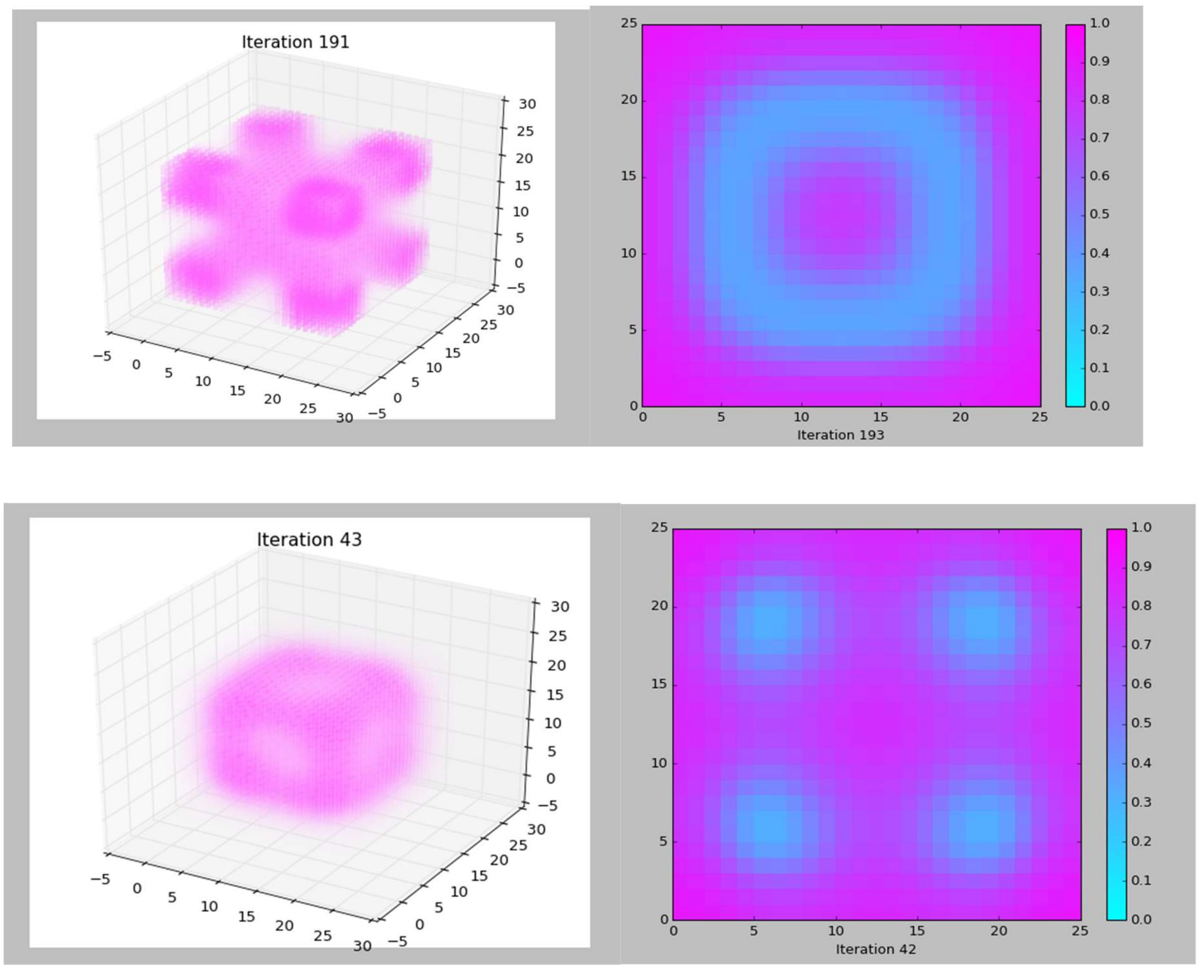

Figure 4 3D simulations along with an XY cross section where Z = 13 (middle Z slice). The upper images had a kill and feed rate of 0.055 and 0.062, respectively, while the lower had 0.032 and 0.062, respectively. In the left images, purple corresponds to concentration of 0 for A in a given cell, while in the right images, cyan does. The iteration numbers are slightly off due to an inability to screencap both at the same time, and the way it was set up, getting the 3D version was tricky as it disappeared after appearing for a short amount of time.

These 3D Simulations are rather promising, as their cross-sections are rather similar to their 2D variants (the major ring staying in the first one, and the separate orbs forming in the second one); however, the 3D version adds some interesting variation to it as well, particularly along the edges of the cubes. However, this is most likely an artifact of the constrained space (as concentrations can exchange along those borders with those on the opposite side). Still, there remains a great distinction between the two different parameters pairs that look quite possible. An important flaw in this visualization technique is the inability to see inside the blob, which is apparent in the first simulation, when comparing to the cross section. The second simulation does not suffer from this issue as it does not have a shell, and in fact, this visualization makes it easy to see that there is a high concentration of A inside.

Discussion

Overall, the implementation of these variants of the Gray-Scott model worked surprisingly well. Developing and testing with these methods increasingly demonstrated the versatility of this single model, as slight changes to the parameters were able to produce quite different results. However, this implementation served to highlight the difficulties of this type of modeling, and allowed me to better understand the limitations of computing in such types of modeling.

Foremost, the sheer computational intensity of these types of method truly struck when attempting to generate a large grid even at the 2D level, and the length of time between iterations made simulations take much longer than anticipated. At the 2D level, this was okay because it was still possible to have a large enough grid to obtain decent results in good time (1 iteration of a 50 x 50 grid takes roughly 1-2 seconds). Though at the 3D level, the tradeoff was too high with a 25 x 25 x 25 grid, and it interfered with the simulation, though it was necessary for this project in order to get simulation results in reasonable time (8-9 seconds per iteration). As not much can be done to affect the overall algorithm, implementing this to run on the GPU would decrease simulation time massively. To fix this issue, trying to integrate CUDA support, a GPU interface with Nvidia graphic cards, would be a nice next step.

The other major complication that I came to appreciate in modeling is the difficulty in transferring from a 2D model to a 3D model. Not only is there a massive increase in the computational costs associated with it, but the visualization aspect becomes much more difficult. For instance, it becomes difficult to determine whether the shapes formed in the 3D version are solid or hollow without having a cross section to guide you. My simple implementation utilizing the 3D scatterplot in matplotlib works alright, but a better implementation can be made that utilizes some actual 3D modeling, such as using openGL. Even slight differences between the 2D and 3D simulations can be difficult to pinpoint, such as the change in seed type from a single point to a cluster of points was tricky as the 2D version has no analogous difficulties, and only experimentation can be used to find the solutions to such issues.

Overall, implementing the different variants of this model served to highlight difficulties of computational modeling in general, but it also served to demonstrate how even simple models can provide greater complexity than expected.

1 Kerr, R. A. *et al.* FAST MONTE CARLO SIMULATION METHODS FOR BIOLOGICAL REACTION-DIFFUSION SYSTEMS IN SOLUTION AND ON SURFACES. *SIAM J Sci Comput* **30**, 3126, doi:10.1137/070692017 (2008).

2 Andrews, S. S., Addy, N. J., Brent, R. & Arkin, A. P. Detailed simulations of cell biology with Smoldyn 2.1. *PLoS Comput Biol* **6**, e1000705, doi:10.1371/journal.pcbi.1000705 (2010).

3 Bartol, T. M. *et al.* Computational reconstitution of spine calcium transients from individual proteins. *Front Synaptic Neurosci* **7**, 17, doi:10.3389/fnsyn.2015.00017 (2015).

4 Johnson, T., Bartol, T., Sejnowski, T. & Mjolsness, E. Model reduction for stochastic CaMKII reaction kinetics in synapses by graph-constrained correlation dynamics. *Phys Biol* **12**, 045005, doi:10.1088/1478-3975/12/4/045005 (2015).

5 Subburaj, Y. *et al.* Bax monomers form dimer units in the membrane that further self-assemble into multiple oligomeric species. *Nat Commun* **6**, 8042, doi:10.1038/ncomms9042 (2015).

6 Liu, X., Welf, E. S. & Haugh, J. M. Linking morphodynamics and directional persistence of T lymphocyte migration. *J R Soc Interface* **12**, doi:10.1098/rsif.2014.1412 (2015).

7 Turk, G. Vol. 25 289-298 (Computer Graphics, 1991).

8 Karst, N., Dralle, D. & Thompson, S. Spiral and Rotor Patterns Produced by Fairy Ring Fungi. *PLoS One* **11**, e0149254, doi:10.1371/journal.pone.0149254 (2016).
9 Komoto, A. & Maenosono, S. Photoinduced fluorescence intensity oscillation in a reaction-diffusion cell containing a colloidal quantum dot dispersion. *J Chem Phys* **125**, 114705, doi:10.1063/1.2338804 (2006).
10 http://www.karlsims.com/rd.html.
11 http://pmneila.github.io/jsexp/grayscott/.