CS279 Final Project

The Gray-Scott Model: An Implementation in Two and Three Dimensions

Luísa Galhardo, Geronimo Garcia

Introduction

Computational models aim to explore biological systems and phenomena. To simulate the dynamics of biological molecules throughout cells, diffusion models use derived deterministic laws to govern the movement of particles. We explored two different single-species particle diffusion models in assignment 3. Through this project, we hope to explore the movement and interaction of two species through the Gray-Scott Diffusion-Reaction Model. The versatility of this model can be appreciated by noticing that slight tweaking of its parameters causes vast differences in results. Specific reaction parameters can result in "Turing patterns", which highly resemble natural patterns and processes such as leopard spots, zebra stripes,[1] and fungi growth[2].

The Gray-Scott Model

The Gray-Scott Diffusion-Reaction Model simulates the movement and interactions of two species, A and B. The concentrations of these species are governed by the following differential equations:

$$\frac{\partial A}{\partial t} = D_A \nabla^2 A - AB^2 + f(1 - A)$$

$$\frac{\partial B}{\partial t} = D_B \nabla^2 B + AB^2 - (f + k)(B)$$

Each term in this equation represents an aspect of the Gray-Scott model[4]:

The diffusion of species A and B is calculated by their respective diffusion rates, D_A and D_B, and the laplacian operator ∇^2 which convolves the values of their neighboring cells with a shared kernel.

The second term is derived from the reaction between species A and B. The reaction between A and B occurs as follows:

$$\mathrm{A} + 2\mathrm{B} \rightarrow 3\mathrm{B}$$

which yields the term AB^2, negative in A's differential equation since movement to the right of this equation reduces the concentration of A, and positive in B's differential equation since the reaction increases the concentration of B.

Lastly, in the Gray-Scott model, particle A is said to have a feed rate of f and particle B is said to have a kill rate of k. These parameters are of particular interest in the Gray-Scott model as small changes generate vastly different patterns which we will discuss further in this report.

We have taken a continuum-based approach to this implementation, so for this model, we can imagine a space divided into a grid, or "cells", where within each cell the concentrations of both species are tracked.

2D Implementation

For our two-dimensional implementation, we used the code from assignment 3 as a starting skeleton for the repeated calculation and display of concentrations in each "cell" involved in our continuum-based model. We altered the algorithm to perform diffusion-reaction of species A and B as related above.

Our algorithm uses two different data structures to separately track the concentrations of A and B at each cell (x, y) in the grid. The grid is initialized with concentration of A = 1, B = 0 at every cell and seeded with B = 1 at the center of the grid. We chose to display the concentration of species B in our visual plot because this is the species of interest when examining the diffusion in the model, considering that A begins spread out with equal concentration and B diffuses out from the seed. The display could, however, be based on the concentration of either species.

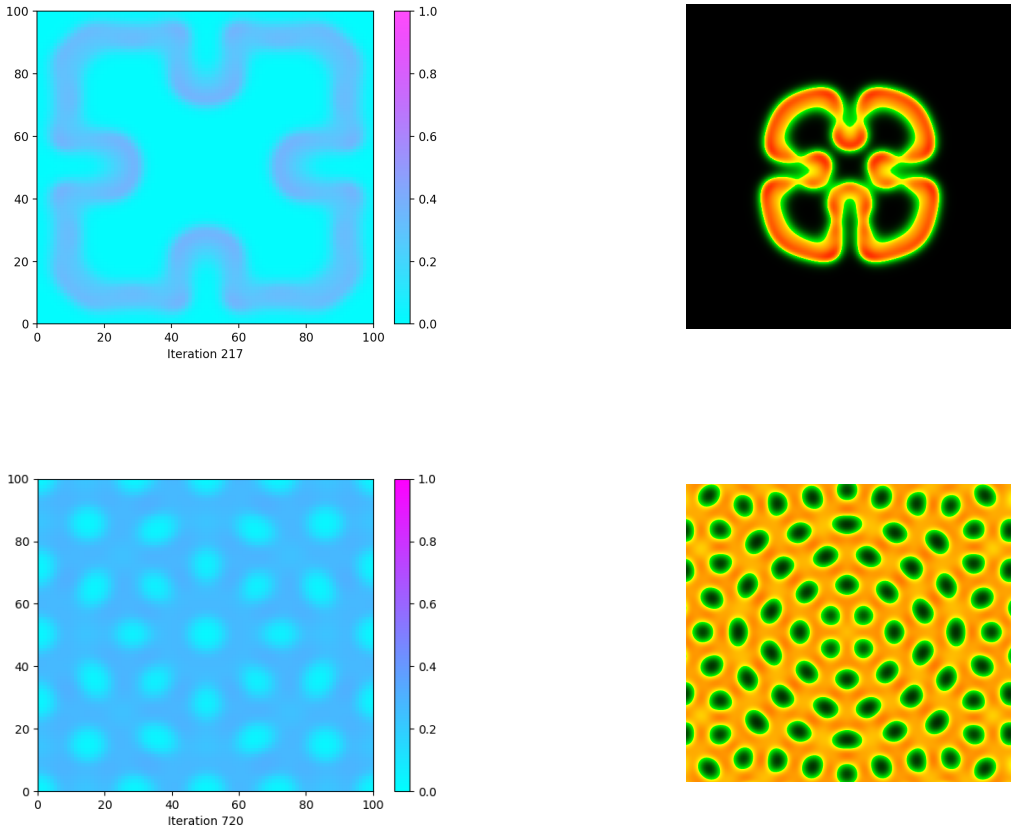
In testing our implementation, we kept all parameters constant except the kill and feed rates. That is, for the entirety of this report, we maintained $D_A = 1.0$ and $D_B = .05$, and chose one convolution matrix to use for each implementation. For this two-dimensional model, we chose the following convolution matrix for the laplacian operator:

$$\begin{bmatrix} .05 & .2 & .05 \\ .2 & -1 & .2 \\ .05 & .2 & .05 \end{bmatrix}$$

We see that this matrix weights adjacent cells more heavily than cells diagonal to the center cell being examined. This is because, in theory, adjacent cells should have more of an effect on the diffusion of the cell we are examining since they have more surface area in common. The values of the matrix add to 0 to conserve the number of particles in the diffusion. In the next section we will see that we chose a similar approach for the 3D convolution matrix.

To test our model, we examined trials where $f = 0.037, k = 0.06$ and where $f = 0.039, k = 0.058$, which

resulted in two distinct patterns, and we compared our results to existing models to ensure correctness:

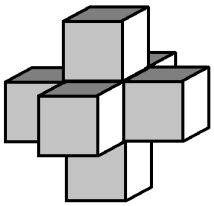


To the left we see first the 2D representation of our Gray-Scott model where $f = 0.037, k = 0.06$ and below that where$f = 0.039, k = 0.058$. To the right are screenshots of the model presented in class run with the corresponding parameters[3]. We believe the discrepancies in pattern can be explained by the different spaces the two models had to diffuse in, our model's being much smaller, but maintain that the general shape and pattern indicates our implementation was successful.

3D Implementation

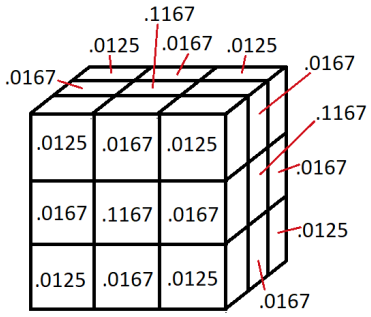
When deciding to implement this diffusion-reaction algorithm in 3D, we chose to extend our 2D implementation with iteration and plotting methods redesigned to accommodate for three dimensions. The differential equations and reactions governing the movement of the species remained the same.

Our algorithm once again uses two different data structures to separately track the concentrations of A and B at each cell, this time at (x, y, z), in the grid. The grid is initialized with concentration of A = 1, B = 0 at every cell and seeded with B = 1 of a center cell and its adjacent cells in the direction of the axes:



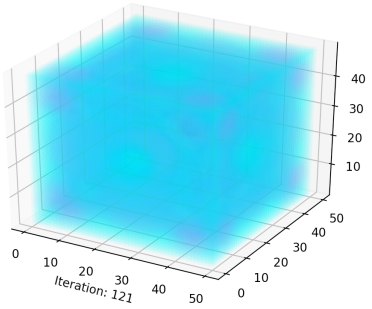
We found that seeding only 1 cell with B, like in the 2D implementation, caused the diffusion to "die", so a bigger seed was necessary. We chose again to display the concentration of B for clarity, but as before, the display could be based on the concentration of either species.

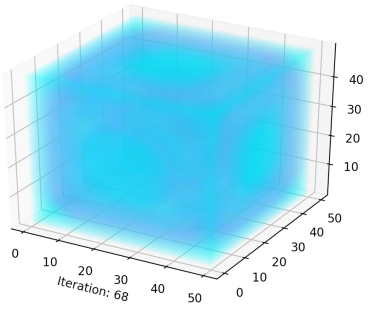
We proceeded with the same diffusion constants as before. For this three-dimensional model, we chose the following 3D convolution matrix for the laplacian operator:



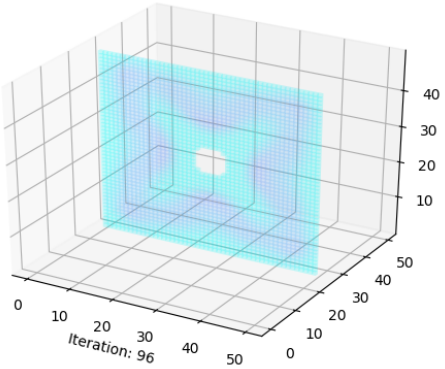
This is a 3x3x3 matrix where the corners are .1/8 = .0125, the cells adjacent to the center cell are .7/6 = .1167, and the rest of the cells, apart from the center one, are .2/12 = .0167. The center cell is -1 just as in the 2D model, so that the matrix sums to 0. We chose the distribution of the weights based on the relationship between weights in the 2D kernel. We see this matrix once again weights adjacent cells more heavily than the others. The corner cells have the least effect on the diffusion term because these cells have very little contact with the center cell being examined.

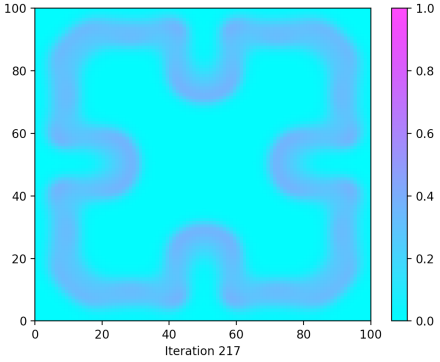
To test our model, we again examined trials where $f = 0.037, k = 0.06$ and where $f = 0.039, k = 0.058$, which resulted in two distinct patterns, and we compared our results to our two-dimensional model. Below we see first the 3D representation of our Gray-Scott model where $f = 0.037, k = 0.06$ and next to that where $f = 0.039, k = 0.058$.

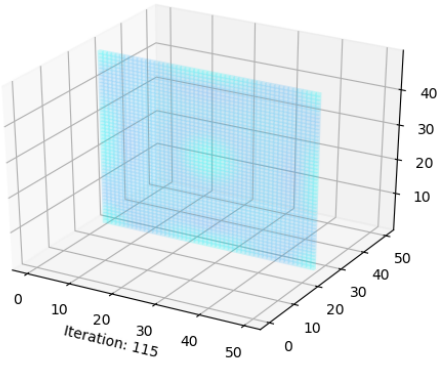


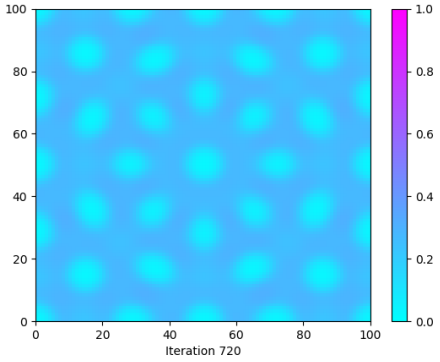


Now to examine correctness of our algorithm, we compared the diffusion process of a slice of the 3D model against our 2D model. The slice of our 3D model is created by solely plotting $y = length/2$ (note that all voxels are still calculated). These images are to the left. To the right are (again) the images from our 2D implementation with these parameters:







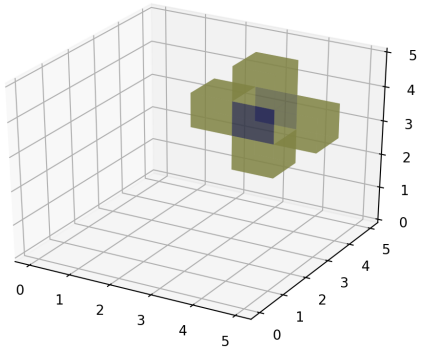


We see that a cross-section of the 3D model looks very similar to the 2D implementation. In the top row, there is a clear flowery/cross shape that develops. The center is "missing" because our model plots a concentration of 0 as no voxel instead of as a color like we see in the 2D colorbar. In the second row, the result is slightly faint, but we can see a hole in the middle and some holes developing at the top, bottom, left, and right of the slice. This leads us to believe that we successfully implemented the diffusion in 3D.

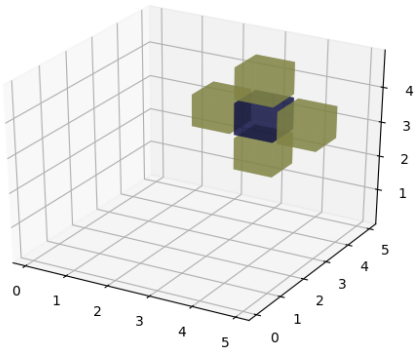
Discussion

We faced many difficulties in representing the three-dimensional model visually. We spent time researching implementation methods that were within our skill sets and were easy to test. We finally decided on using voxels in matplotlib, which we could color using a defined colormap and also make translucent to facilitate viewing in 3D. Still, we had to debug how we implemented this visualization so that the representation was clear and accurately reflected the concentrations in our data structure.

Initially, individual voxels with more than one face color would sometimes appear, leaving the appearance of a "floating" face of a certain color, which posed a problem as each voxel should have uniform coloring to represent the concentration within that voxel. We discovered that faces of adjacent voxels touched or shared the same space. Therefore, face colors could be overwritten at this boundary so that some voxel face colors seemed to "disappear" when surrounded by other voxels, as shown in the following picture:



The voxels needed to be separated so that faces would not overwrite each other. We encountered code [4] that provides an "explode" function that gives each voxel buffer space in all dimensions to separate them from other voxels. Once the "explode" function was applied, faces no longer shared spaces and face colors were not overwritten:



This provided a much more intuitive and pleasing visualization of the concentrations.

To better visualize the data, further efforts could choose to implement the three-dimensional display of these algorithms through OpenGL. We were unsure if this would be a worthwhile investment in understanding the model, but a higher and finer graphical representation can provide opportunity for better investigation of parameters. Another idea is to implement this as a particle-based model instead of a continuum model, perhaps through a scatterplot in matplotlib or some more involved method in OpenGL or webGL.

More ideas for investigation in these models is to explore the roles of the weights in the Laplacian in resulting patterns. In our two-dimensional implementation, we used a known 2D Laplacian convolution kernel [5], but our three-dimensional model uses our own 3D kernel. Exploring different distributions of the kernel's weights could lead to a more optimal algorithm for simulating this diffusion reaction in 3D space.

Lastly, efforts could be made to optimize space and time requirements, as both implementations take a longer than desired time to run.

We leave this project with a better understanding of how the Gray-Scott Diffusion-Reaction works and the versatility of its parameters. We also came to appreciate the effort required to extend a two-dimensional model into three-dimensions.

Contributions

Luísa Galhardo wrote initial 2D implementation, debugged and brainstormed 3D implementation, and contributed to write-up.

Geronimo Garcia debugged and brainstormed 2D implementation, wrote initial 3D implementation, and contributed to write-up.

References

[1] J.D. Murray. *Mathematical Biology.* Springer-Verlag, New York, 1989.

[2] Karst, N., Dralle, D. and Thompson, S. *Spiral and Rotor Patterns Produced by Fairy Ring Fungi.* PLoS One 11, e0149254, doi:10.1371/journal.pone.0149254 (2016).

[3] github user pmneila. *Reaction diffusion system (Gray-Scott model)*, `https://pmneila.github.io/jsexp/grayscott/`

[4] matplotlib documentation. *Reaction Diffusion: The Gray-Scott Algorithm*, `https://matplotlib.org/gallery/mplot3d/voxels_numpy_logo.html`

[5] Cope, Greg. *Reaction Diffusion: The Gray-Scott Algorithm*, `www.algosome.com/articles/reaction-diffusion-gray-scott.html`