A Couple of Implementations of Reaction Diffusion Models

Olivia Lundelius

Motivation

One of Alan Turing's somewhat lesser known contributions to the sciences is his article titled The Chemical Basis of Morphogenesis[1]. In it, Turing described a set of mathematical models known as reaction-diffusion systems which correspond to many physical phenomena, generating beautiful and intricate patterns, many of which are sometimes likened to Keith Haring paintings. Reaction diffusion is a topic of interest in both biology and computer graphics, so this paper describes my work in simulating reaction diffusion systems on a 2D plane and across mesh surfaces using vertices as discretized points across a mesh.

Related Work

Though the title of the paper includes the word "morphogenesis," Turing's paper describes how patterns found in nature, such as stripes, texture, and spotting, are a result of chemical diffusion during embryonic phases of an organism's life (which certainly includes morphology, but is even broader in its application than just that). Turing's use of the word morphogenesis refers specifically to his concept of "morphogens," which he coined to describe the catalysts and instructions required to propagate patterns across a surface or throughout a tissue. These processes he described later came to be known as the reaction-diffusion system, and they are key to a theory of morphogenesis by the same name. They can be used to describe a wide range of wave-like phenomena and self-organized patterns, including particle simulation. The reaction-diffusion theory of morphogenesis focuses on the result of local chemical reactions and diffusion of substances over surfaces to explain such topological formations and breaks from natural symmetry[2]. While this contribution by Turing is certainly of great use and interest to fields of biology and botany, it would be more accurate to call it a

contribution to what we now call pattern formation, a more abstracted field. It is the result of Turing's numerous interactions with scientists from a variety of fields, from chemistry and zoology to mathematics and physics, to produce a set of equations that unify these disciplines. These models still inform a number of biological models of pattern formation today, including local autoactivation-lateral inhibition, which describes vertebrate limb pattern formation[3].

There have been many implementations of reaction diffusion systems, one of the more early ones by Turk[4]. Turk's 1991 simulation of reaction diffusion allowed him to create a variety of patterns both on grid surfaces in 2D as well as apply those processes to 3D meshes. His method for discretization of the grid involved drawing a Voronoi graph of the model's vertices and edges, and using those cells for iteration. Application to natural systems are also a subject of interest, including in the modeling of seashells[5]. Fowler, Meinhardt, and Prusinkiewicz's early use of the reaction diffusion model (also called the activator-inhibitor model) were used to accurately generate colored banding on the shells of various sea snails. More unusual applications have also been made, including using reaction diffusion to guide deformation of a thin shell in procedural modeling[6]. Gingras and Kry stepped through reaction diffusion on a triangulated mesh using vertices as the discretized points for calculating chemical concentrations. These concentrations were then used to apply pressure in the direction of a face's normal to deform the mesh, simulating the localized growth of tissues such as the early development of plant leaves.

Implementation

My first implementation of reaction diffusion was the application of color to a two dimensional grid surface in order to get the reaction diffusion system working properly to demonstrate its continuation and interactions across a surface over time. This was first attempted in R and later moved to Python, and made use of numpy for generating matrices and Python Imaging Library for saving output images.

Mathematically, two chemical reaction-diffusion systems themselves are semi-linear parabolic partial differential equations of the form

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

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

in which A and B are chemical concentrations, D_A and D_B are the diffusion rates for A and B respectively, f is the rate at which chemical A is added, and k is the rate at which chemical B is removed or dissipates. The first terms diffuse the chemicals across a surface by making the grid cell more like surrounding cells. The second terms allow for a reaction in which two parts chemical B cause one part chemical A to turn into chemical B. The feed term $f(1 - A)$allows for consistent addition of chemical A while correcting to ensure A never goes above 1 (100% concentration), while the kill term $(f + k)B$ removes chemical B at a continuous rate.

Functions for iterating over the reaction diffusion equations and approximating the laplacian were separated to allow for easier modification to apply to different systems. Chemical concentrations for chemicals A and B were represented as 2D matrices with each pixel of the output display corresponding to one grid cell entry in the 2D matrix, with A initially set to all 1s and B set to mostly 0s with some 1s. The matrices were iterated over to apply the reaction diffusion equations to the entire grid at a time, placing results in a new pair of A and B matrices. In approximating the laplacian, chemical concentration values in a 3-by-3 submatrix were weighted with the following convolution kernel matrix

$$\begin{pmatrix} 0.05 & 0.2 & 0.05 \\ 0.2 & -1 & 0.2 \\ 0.05 & 0.2 & 0.05 \end{pmatrix}$$

Indices were additionally adjusted at the edges to allow for the patterns to wrap around at the sides, top, and bottom of the grid to prevent the need to change the convolution matrix for edge cases. After iterating, the matrix of chemical B's concentrations were converted to an image and saved, with greater concentrations of B appearing white and lower concentrations appearing black. Parameters for f and k values, number of iterations, and number of images can be passed as arguments when running the program, as well as the square size in pixels of the resulting images. The number of images produced and number of iterations between each generated image can also be passed as parameters.

This script was then modified to work with triangulated matrices in an attempt to allow reaction diffusion to be applied to a range of 3D models. Using Qingnan Zhou's PyMesh library[7], a mesh can be either specified within the program or generated using PyMesh to be used in simulating reaction-diffusion. The matrices for A and B were converted to arrays corresponding to the mesh's indexed vertices, and the calculation of the laplacian was modified to use either PyMesh's built in Laplace-Beltrami calculation across the entire mesh, or an equally weighted method of finding the difference between the averaged concentration values of connected vertices and the central concentration value. Iterations are then opened as individual plotly graphs to allow for examination of the full surface of each iteration. Color is applied based on the concentrations of B at the vertices and is interpolated across mesh faces.

Challenges

Multiple avenues were explored before eventually settling on the language and libraries that were chosen, including using R formulas to initially try and model these systems. Using R formulas turned out to be more challenging than expected and also led to the realization that my implementation would be more difficult to convert to different languages or make modifications, so I swapped to stepping through the reaction diffusion equations more manually and went to use Python instead, in which I was able to make my 2D reaction diffusion turing patterns. In

working on the 3D surface implementation of this project, there were many struggles with installing the PyMesh library on my windows machines. Eventually, after much contention and half hearted attempts to restart once again in C++ and OpenGL, I swapped to using a 2013 Macbook Pro that finally allowed me to install and begin using the library after numerous updates to MacPorts, XCode, my operating system, and several scattered packages necessary to running PyMesh. These issues left me with less time than I would like to work on this part of my project, but I still managed to get some rudimentary pattern generation over surfaces in 3D.

Results

A variety of turing patterns can be produced with the resulting script, and are accurate to the kinds of patterns produced by the given input parameters. Not being parallelized, the program runs rather slowly, with a single iteration taking on average 0.127 seconds over a 100 by 100 pixel image. Figures 1, 2, and 3 each consist of six 100 by 100 pixel images, with 750 iterations between each image. Each image took on average 86.945 seconds to be generated.

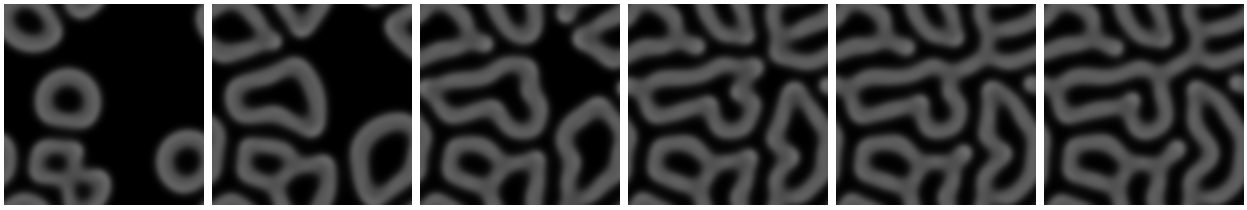

***Fig. 1*:** f = 0.055, k = 0.062

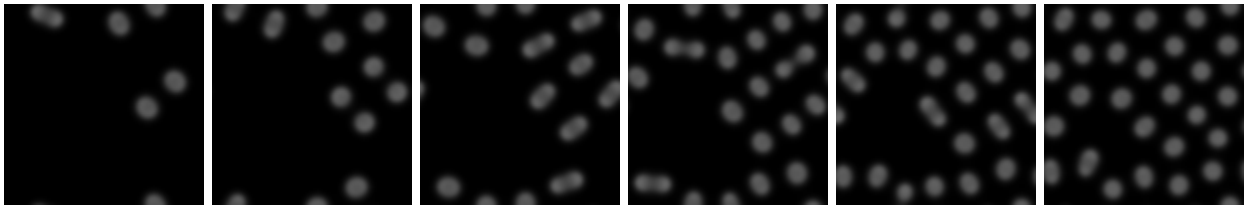

***Fig. 2*:** f = 0.0367, k = 0.0649, “mitosis”

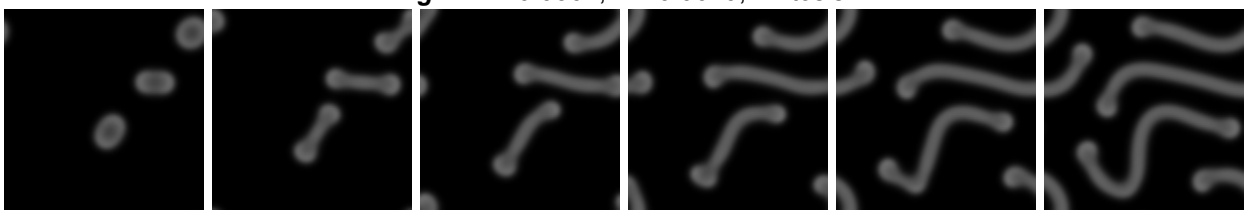

***Fig. 3*:** f = 0.055, k = 0.064

The pattern generation over 3D surfaces still is not fully working properly, but some Turing-like patterns are still visible when the B matrix is not initialized with any starting 1 points, and the concentrations continue to decline. Currently, the concentrations of A and B tend to oscillate greatly from iteration to iteration until eventually reaching Python's overflow. At first I suspected it may be the result of incorrect Laplacian calculation, but using various different methods of approximation continue to result in the strange behavior. Due to using vertices instead of individual pixels or some other, much finer method of discretizing the mesh, the running times for producing images were not as bad as they could have been, but this comes at the cost of relying on the original mesh to be triangulated relatively evenly.

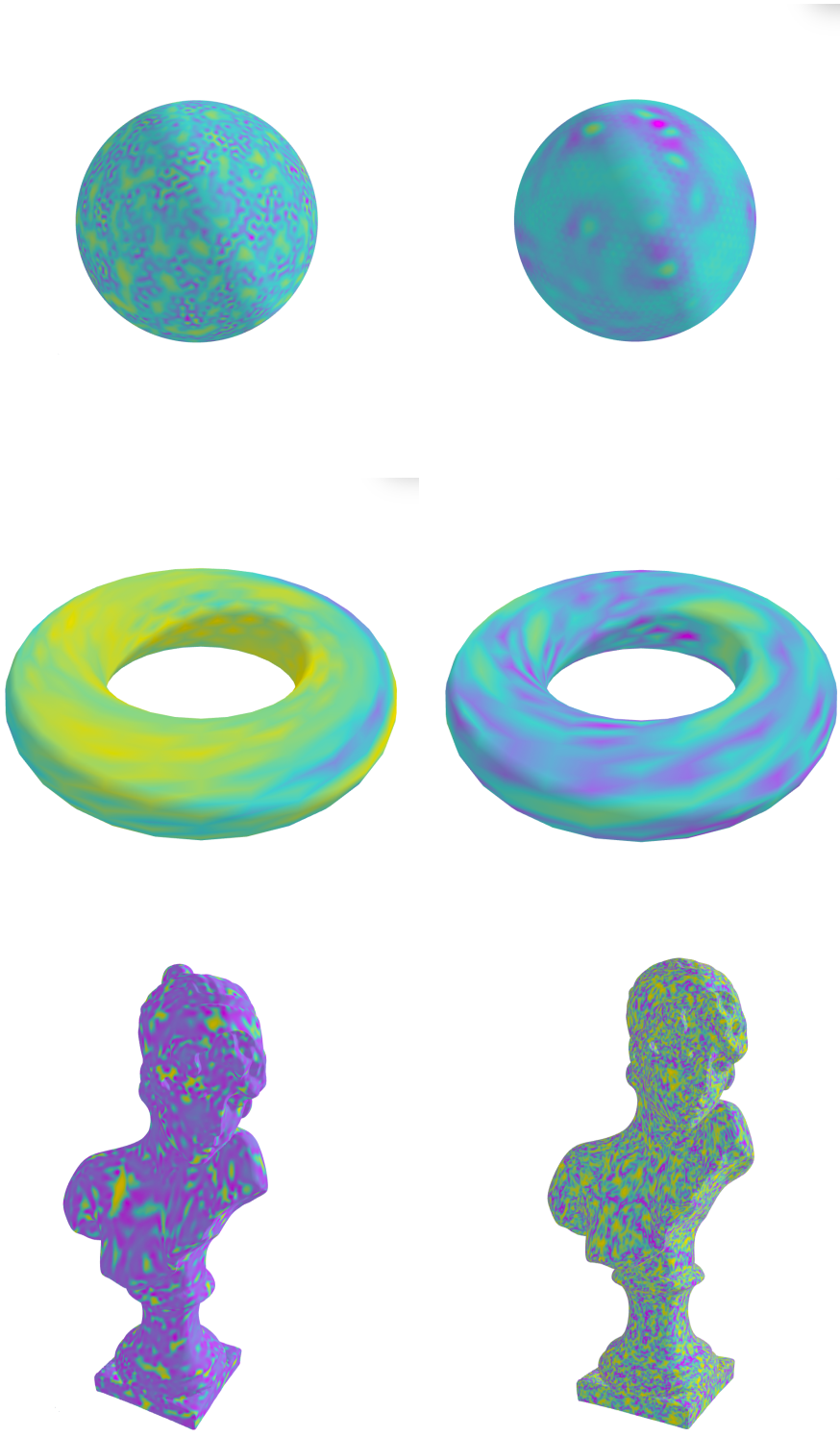

Fig. 4: some interesting patterns generated on meshes, from top to bottom: icospheres generated using PyMesh, quad-mesh toruses, and triangulated busts of Sappho.

Conclusion and Future Work

Fixing the reaction diffusion over a mesh would allow for better experimenting with different meshes to determine what kinds of meshes work well with this method and what modifications to a mesh may be needed to improve visual effects. Parallelization of both the 2D and mesh programs would likely speed up processing time greatly. Use of Python's pool function from the multiprocessing library may be useful, or using PyOpenGL to allow for computation on the GPU. The program could also be modified to use voxels in a 3D grid system as an extension of the 2D program, functioning as a kind of procedural modeling, allowing even more interesting and organic shapes to arise. A potential further extension of this project could be to implement the reaction-diffusion system as a solid texture, as described in Perlin's paper An Image Synthesizer[8]. This may be particularly challenging as the reaction-diffusion equation is usually a descriptor of a topological model for pattern formation, but it may be possible to "cheat" the system by extending it further into 3 dimensions, possibly by a point-cloud system[9] and applying it to the solid as a whole. This would allow the pattern to relatively easily be made topological without trying to rely on color as a height map, and trying to deal with subdivision, extrusion, and similar deformative operations. However, this may result in iterations that don't make sense visually as a result of computation of the reaction diffusion equations happening "inside" the mesh. Final resting-state results may still bear semblance to the two-dimensional counterpart.

The two dimensional program works rather well, and while there is certainly room for improvement for my mesh script, the fundamentals are still there and will serve as a useful basis for continuation of this project.

Bibliography

(1) Turing, A.M. “The Chemical Basis of Morphogenesis.” *Philosophical Transactions of the Royal Society of London*. Series B, Biological Sciences, Vol. 237, No. 641. (Aug. 14, 1952), pp. 37-72

(2) Nanjundiah, Vidyanand. “Alan Turing and ‘The Chemical Basis of Morphogenesis’.” *Morphogenesis and Pattern Formation in Biological Systems*, pp.33-44.

(3) Economou, Andrew D.; Green, Jeremy B.A. “Modelling from the experimental developmental biologist’s viewpoint.” *Seminars in Cell and Developmental Biology*. 2014 Nov; 0: 58–65.

(4) Turk, Greg. “Generating textures on arbitrary surfaces using reaction-diffusion.” *Computer Graphics SIGGRAPH*. Pgs 289–298, New York, NY, USA, 1991. ACM Press.

(5) Fowler, Deborah R; Meinhardt, Hans; Prusinkiewicz, Przemyslaw. “Modeling seashells.” *Computer Graphics SIGGRAPH*. Pgs 379–387, New York, NY, USA, 1992. ACM Press.

(6) Gingras, Charles; Kry, Paul G. “Procedural Modelling with Reaction Diffusion and Growth of Thin Shells.” *Proceedings of Graphics Interface 2019*: Kingston, Ontario, 28 - 31 May 2019

(7) Zhou, Qingnan. *Pymesh*. https://github.com/PyMesh/PyMesh.

(8) Perlin, Ken. “An Image Synthesizer.” *ACM SIGGRAPH Computer Graphics*. July 1985, Vol. 19, No. 3.

(9) Macdonald, Colin B.; Merriman, Barry; Ruuth, Steven J. “Simple computation of reaction-diffusion processes on point clouds.” *Proceedings of the National Academy of Sciences*. May 2013, 110(23).