A Finite Element Approach to Reaction-Diffusion Systems

Gavin Engelstad*

gengelst@macalester.edu.

Fall 2024

Abstract

TBD

1 Introduction

Reaction-diffusion systems model the spatiotemporal behavior of various chemical and physical systems of one of more components that react with each other and diffuse across space. Chemical reactions [25, 8], the human nervous system [5], population dynamics [2], and the patterns that show up on animal's skins [23, 4] can all be described using versions of a reaction-diffusion system. Within a system, reactions turn one substance into another as diffusion causes substances to spread out [13]. The solutions to reaction-diffusion systems presents interesting patterns (dubbed "Turing Patterns" [24]), moving fronts, and oscillations [22, 19, 23].

Because of the wide range of applications reaction-diffusion systems have, it is important to have accurate and efficient computational methods to solve them. The main challenge to solving the reaction-diffusion systems is the nonlinear reaction term [18, 14]. This term drives the asymptotic behavior and stability of the system [18], but it also introduces additional complexity, especially for methods that abuse local, linear approximations of the system to solve it. Finite differences [9], spectral [1, 3], and analytic [21] approaches can solve this problem with various degrees of

*Replication code available at `https://github.com/GavinEngelstad/Reaction-Diffusion-FEM`.

computational efficiency and accuracy, but are limited to only work for certain reaction-diffusion systems or domain shapes.

This paper presents an approach to solving reaction-diffusion systems that uses the finite element method (FEM). The method in this paper is heavily based on the mathematical approach of [20] and [12]. I utilize finite elements to discretize the space derivative and an implicit (backwards) Euler method to discretize the time derivative. Using a fine enough triangular mesh, the method is numerically stable and can be applied to a wide range of domains, including standard disks and rectangles, more complicated maze-like structures, and along the surface of 3D shapes.

2 The Finite Element Method

2.1 A General Reaction-Diffusion Equation and its Weak Form

The general form of a reaction-diffusion system is

$$\partial_t \mathbf{u} = \Gamma \nabla^2 \mathbf{u} + \mathbf{R}(\mathbf{u})$$

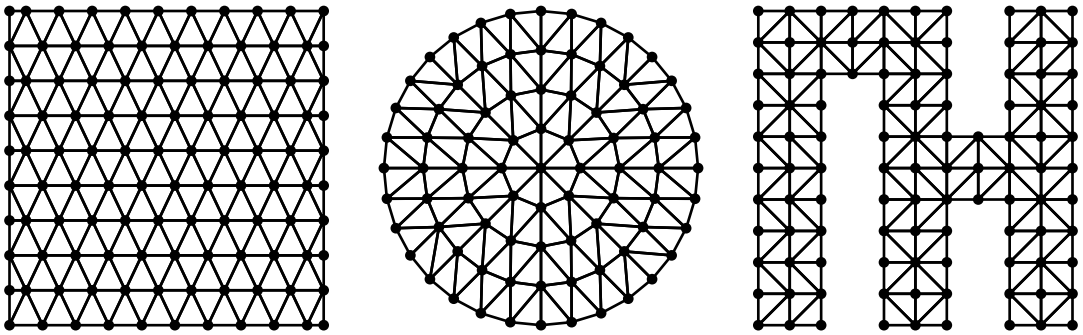
where $\mathbf{u}(t, x, y)$ is a vector function describing the concentration of the reactants, Γ is a diagonal matrix of diffusion coefficients, and $\mathbf{R}(\mathbf{u})$ is a potential nonlinear function describing the reactions between the reactants [14]. I solve the system on the domain Ω which I assume the PDE has Neumann boundary conditions with derivative 0 for simplicity. The method could be extended to Neumann conditions with nonzero gradient across the boundary or Dirichlet conditions, but this is beyond the scope of this paper. Assuming the systems contains N total reactions, each equation in the system is

$$\partial_t u_n = \gamma_n \nabla^2 u_n + r_n\left(\{u_m\}_{m=1}^N\right), \quad n \in \{1, 2, \ldots, N\}.$$

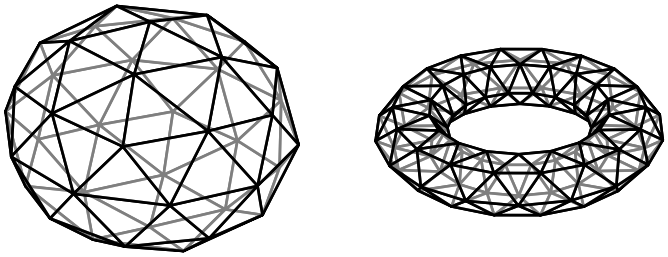
The FEM solves for a solution to the weak form of the PDE [7]. The weak form is given by multiplying by some function v and integrating across the whole domain, which gets

$$\int_\Omega v \partial_t u_n dA = \int_\Omega v\left[\gamma_n \nabla^2 u_n + r_n\left(\{u_m\}_{m=1}^N\right)\right] dA.$$

Figure 2.1: Triangulations of different domains



(a) 2D Domains



(b) 3D Surface Domains

By the product rule for gradients and Neumann boundary conditions, we get

$$\int_\Omega v\gamma_n \nabla^2 u_n = -\gamma_n \int_\Omega \nabla v \cdot \nabla u_n dA.$$

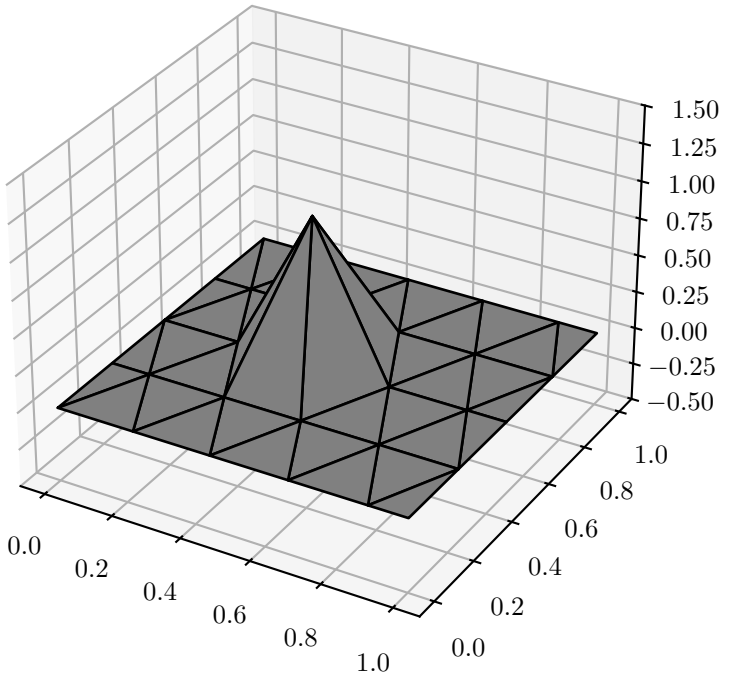
Therefore, the weak form is

$$\int_\Omega v\partial_t u_n dA = -\gamma_n \int_\Omega \nabla v \cdot \nabla u_n dA + \int_\Omega v r_n \left(\{u_m\}_{m=1}^N\right) dA.$$

2.2 Discretization

To discretize the spacial derivative, I triangulate Ω. Figure 2.1 gives examples of what potential triangulations look like for a variety of domains. Panel 2.1a shows this discretization for 2D domains

Figure 2.2: The function ψ_i at a vertex



including a square, disk, and maze-like structure and panel 2.1b shows this discretization along the surface of 3D objects, including a sphere and a torus.

For each vertex v_i in the triangulation, define a function ψ_i such that

$$\psi_i(v_i) = 1$$
$$\psi_i(v_j) = 0, \quad j \neq i$$

and $\nabla\psi_i$ is constant along a triangle T. An example of ψ_i for a vertex in my triangulation is in Figure 2.2. These functions will approximate the weak form of the PDE and the nonlinear reactions.

To discretize u_n, let $u_{n,j}(t)$ approximate $u(t, v_i)$. Then, we have

$$\hat{u}_n(t, x, y) = \sum_j \psi(x, y) u_{n,j}(t) \approx u(t, x, y).$$

Similarly, we can approximate r_n as

$$\hat{r}_n(t,x,y) = \sum_j \psi(x,y) r_n\left(\{u_{m,j}(t)\}_{m=1}^N\right) \approx r(\{u_m(x,y,t)\}_{m=1}^N).$$

This approximation requires the solution for u_n and r_n to be adequately smooth across space and the triangulation to have a fine enough mesh. I analyze these assumptions in Section TBD by solving systems with increasingly fine discretizations to get closer to the exact answer. Using these approximations and $v = \psi_i$ in the weak form gets

$$\sum_j \partial_t u_{n,j} \int_\Omega \psi_i \psi_j dA = -\gamma_n \sum_j u_{n,j} \int_\Omega \nabla\psi_i \cdot \nabla\psi_j dA + \sum_j r_n\left(\{u_{m,j}(t)\}_{m=1}^N\right) \int_\Omega \psi_i \psi_j dA$$

which represents the spatially discretized version of the system.

The temporal discretization is chosen to handle potential nonlinearities in the reaction function as well as maximize numerical stability. Specifically, I combine an explicit (forwards) Euler method to discretize the reaction step and an implicit (backwards) Euler method to discretize the diffusion step [20]. The explicit Euler step evaluates the reaction with known quantities and avoids nonlinearities when setting up a linear system, and the implicit Euler step maximizes the numerical stability of the solution [6]. Letting $u_{n,j}^t = u_{n,j}(t)$ at some discrete time step t, the time-discretized equation is

$$\sum_j u_{n,j}^t \left(\int_\Omega \psi_i \psi_j dA + \Delta t \gamma_n \int_\Omega \nabla\psi_i \cdot \nabla\psi_j dA\right) = \sum_j \left(u_{n,j}^{t-\Delta t} + \Delta t r_n\left(\{u_{m,j}^{t-\Delta t}\}_{m=1}^N\right)\right) \int_\Omega \psi_i \psi_j dA.$$

2.3 Setting up a Linear System

To solve the time-discretized equation, I turn it into a linear algebra problem. I define the damping matrix $\mathbf{D}$ so that

$$d_{i,j} = \int_\Omega \psi_i \psi_j dA$$

and the stiffness matrix $\mathbf{S}$ so that

$$s_{i,j} = \int_{\Omega} \nabla\psi_i \cdot \nabla\psi_j dA.$$

Since $d_{i,j} = 0$ and $s_{i,j} = 0$ whenever v_i and v_j don't share a triangle, the damping and stiffness matrices have a sparse structure that makes computation with them fast even on very fine triangulations of Ω. Appendix A derives expressions for $d_{i,j}$ and $s_{i,j}$ in cases where v_i and v_j share at least one triangle.

Then, defining $\mathbf{u}_n^t$ as the vector with entries $u_{n,j}^t$, the equation becomes the linear system

$$\left(\mathbf{D} + \Delta t\gamma_n \mathbf{S}\right)\mathbf{u}_n^t = \mathbf{D}\left(\mathbf{u}_n^{t-\Delta t} + \Delta t r_n\left(\{\mathbf{u}_m^{t-\Delta t}\}_{m=1}^N\right)\right)$$

which allows me to solve for the solution at t given a solution at time $t - \Delta t$. The function r_n is vectorized and, because $\mathbf{D} + \Delta t\gamma_n\mathbf{S}$ is sparse, symmetric, and positive-definite, I use the conjugate gradient method to solve for $\mathbf{u}_n^t$ [16]. Together, this allows me to quickly solve for each time step.

Therefore, to solve the system, I start with some initial condition $\mathbf{u}_n^0$ for each n. In the sections that follow, this initial condition is chosen to be random noise except when noted otherwise. Then, I solve the linearized version of the PDE for each n to iterate to the next time step. Repeating this process to some final T allows me to efficiently and accurately solve the reaction-diffusion PDE on any triangulated domain.

3 Solving a System

3.1 A Baseline System

For this paper, I solve the system in [17, 10]. By replacing the reaction function, the method can be extended to any sufficiently smooth reaction diffusion system. This includes Turing's initial version [23, 4] and other activator-inhibitor equations [11, 15] which it has been tested against. The system

I solve is given by

$$\frac{\partial u}{\partial t} = \gamma_u \nabla^2 u + k_1 \left(v - \frac{uv}{1+v^2} \right)$$
$$\frac{\partial v}{\partial t} = \gamma_v \nabla^2 v + k_2 - v - \frac{4uv}{1+v^2}.$$

Like [10], I fix $D_u = 1$ and $k_2 = 11$ in all simulations. I vary D_v and k_1 across simulations to create different Turning patterns, but we always have $D_v < D_u$. The time step is set to $\Delta t = 0.01$, and all systems are iterated until $t = 1,000$, at which point they all have reached a steady state. I triangulate so that there are 250 points per square unit of the domain I solve the PDE on. The plotted solutions will be of u, although v makes a similar pattern.

3.2 Solutions

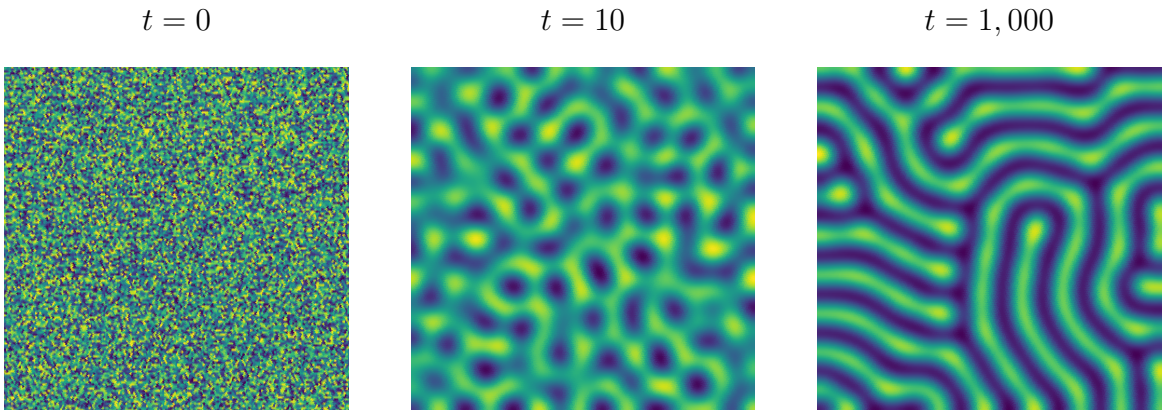
Figure 3.1 plots the solution to the reaction diffusion system at different time steps on a square and circular domain. The left panel plots the initial condition, the middle panel plots an intermediate step during the reaction, and the right panel plots the system at steady state. I set the initial condition to random noise. At $t = 10$, the reaction starts to progress, and we see patterns and 'hotspots' emerge. The patterns that exist, however, lack clear definition. In the steady state, we see well-defined lines that form clear Turing patterns.

At each time step, the system makes similar shapes on the circular and square domains. The intermediate step patterns both have similarly sized hotspots, and the Turing patterns that develop have a similar structure being made of curved-lines with similar width. I explore this further in Section 4 with more complicated domains and arrive at the same result.

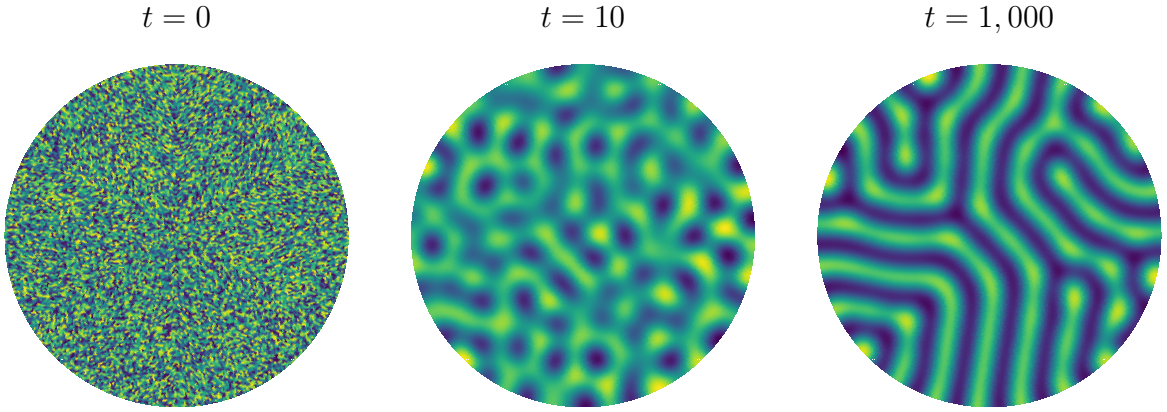
3.3 Parameters

4 More Complex Domains

Figure 3.1: Solutions to the reaction diffusion system



(a) 10×10 square domain



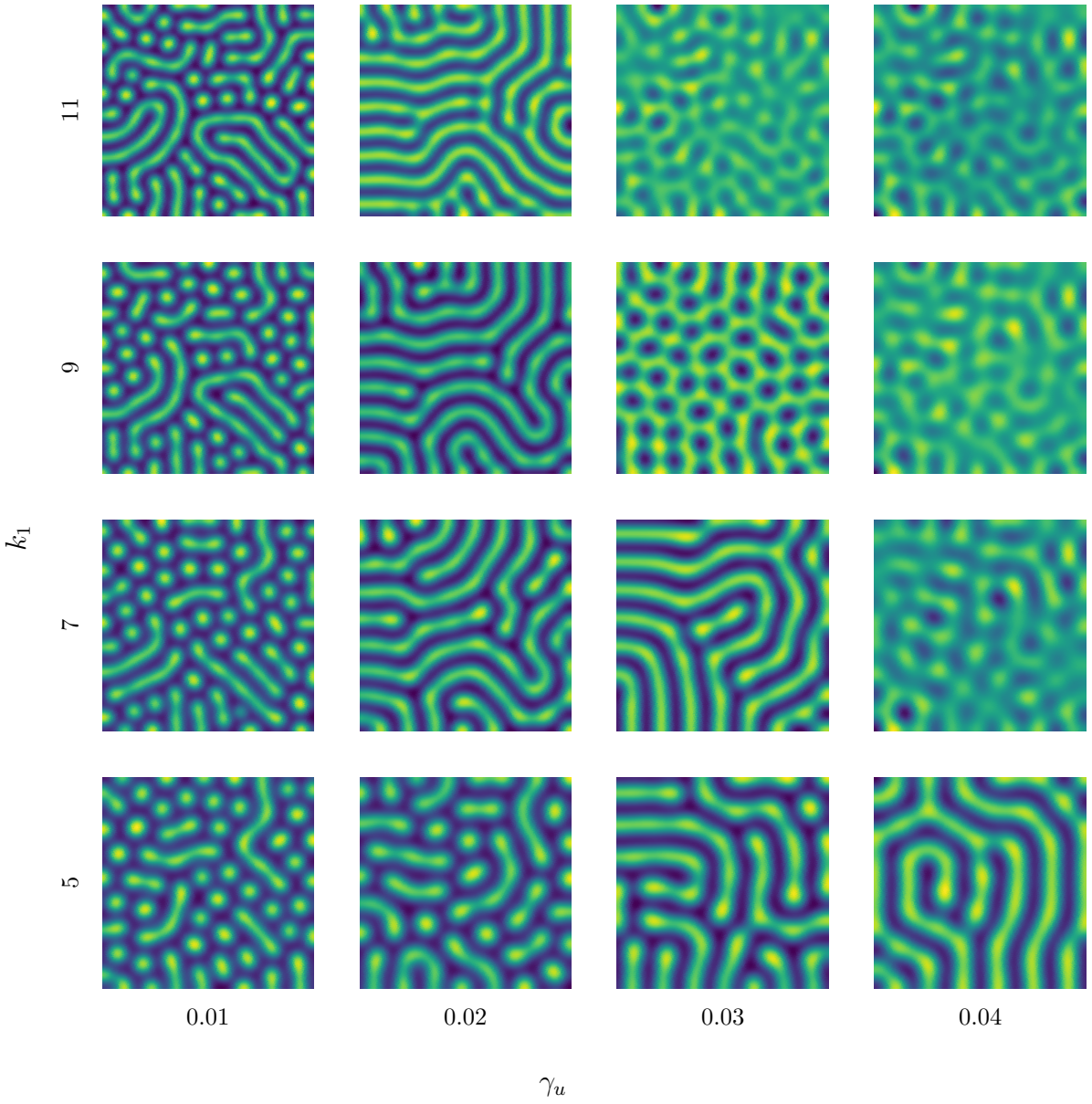
(b) Radius 5 circular domain

References

[1] Alfonso Bueno-Orovio, David Kay, and Kevin Burrage. "Fourier spectral methods for fractional-in-space reaction-diffusion equations". In: *BIT Numerical mathematics* 54 (2014), pp. 937–954.

[2] Ryan St Clair, Andrew Nevai, and Richard Schugart. "A reaction-diffusion model for population dynamics in patchy landscapes". In: *Journal of Differential Equations* 405 (2024), pp. 247–286.

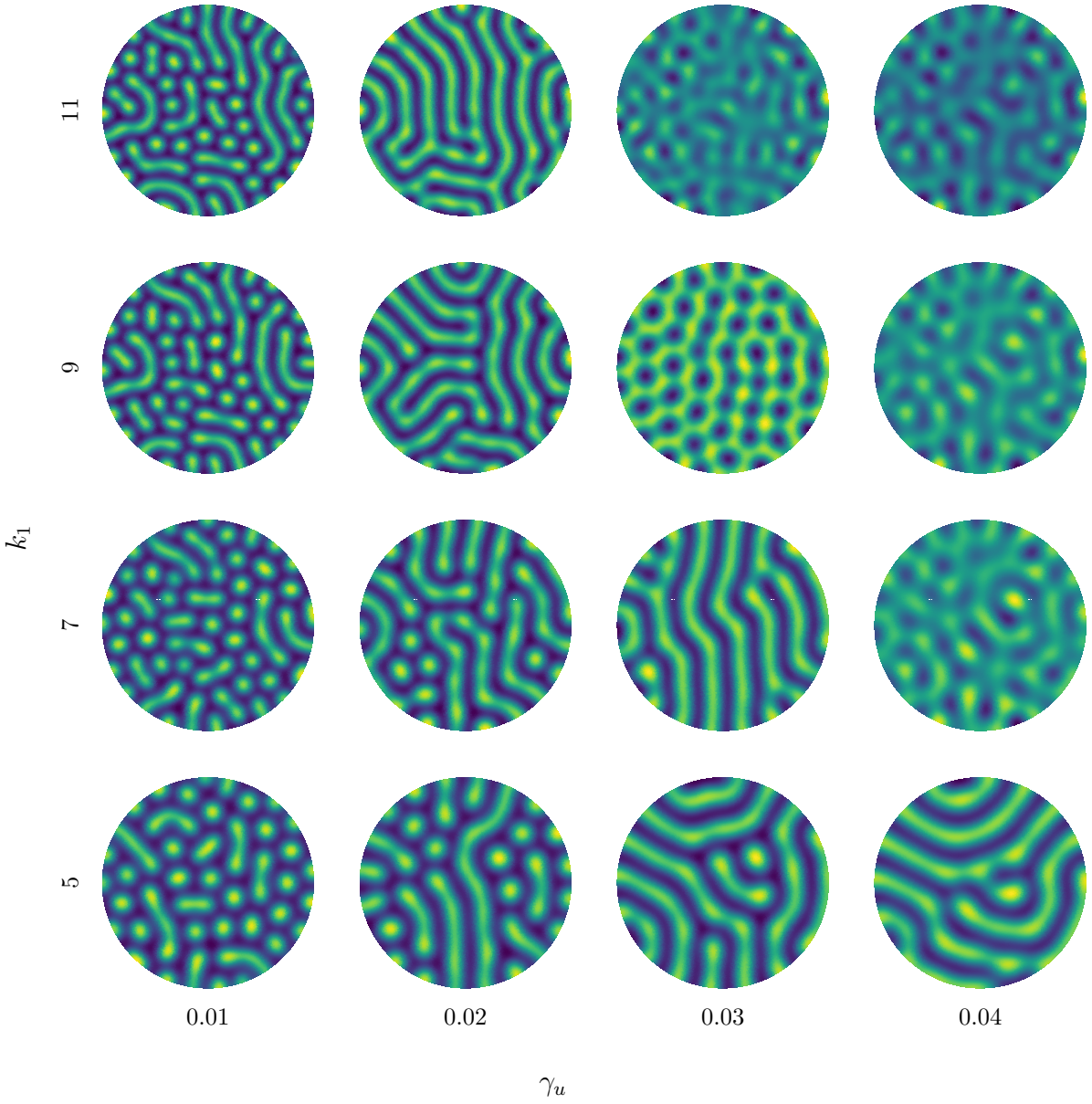
Figure 3.2: 10×10 square steady states for different γ_v and k_1



[3] Richard V Craster and Roberto Sassi. "Spectral algorithms for reaction-diffusion equations". In: *arXiv preprint arXiv:1810.07431* (2018).

[4] Marcelo De Gomensoro Malheiros, Henrique Fensterseifer, and Marcelo Walter. "The leopard never changes its spots: realistic pigmentation pattern formation by coupling tissue growth with reaction-diffusion". In: *ACM Transactions on Graphics (TOG)* 39.4 (2020), pp. 63–1.

Figure 3.3: Radius 5 circle steady states for different γ_v and k_1



[5] Richard FitzHugh. "Impulses and physiological states in theoretical models of nerve membrane". In: *Biophysical journal* 1.6 (1961), pp. 445–466.

[6] Gerald B Folland. *Introduction to partial differential equations*. Princeton university press, 2020.

[7] B.G. Galerkin. *Rods and Plates: Series in Some Questions of Elastic Equilibrium of Rods and Plates*. National Technical Information Service, 1968. URL: https://books.google.com/books?id=gXLCHAAACAAJ.

[8] Michael D Graham, Samuel L Lane, and Dan Luss. "Temperature pulse dynamics on a catalytic ring". In: *The Journal of Physical Chemistry* 97.29 (1993), pp. 7564–7571.

[9] David Hoff. "Stability and convergence of finite difference methods for systems of nonlinear reaction-diffusion equations". In: *SIAM Journal on Numerical Analysis* 15.6 (1978), pp. 1161–1177.

[10] Darae Jeong et al. "Numerical simulation of the zebra pattern formation on a three-dimensional model". In: *Physica A: Statistical Mechanics and its Applications* 475 (2017), pp. 106–116.

[11] Amit N Landge et al. "Pattern formation mechanisms of self-organizing reaction-diffusion systems". In: *Developmental biology* 460.1 (2020), pp. 2–11.

[12] Jens Lang and Artur Walter. "A finite element method adaptive in space and time for nonlinear reaction-diffusion systems". In: *IMPACT of Computing in Science and Engineering* 4.4 (1992), pp. 269–314.

[13] Angran Li et al. "Reaction diffusion system prediction based on convolutional neural network". In: *Scientific reports* 10.1 (2020), p. 3894.

[14] RH Martin Jr and Michel Pierre. "Nonlinear reaction-diffusion systems". In: *Mathematics in science and engineering*. Vol. 185. Elsevier, 1992, pp. 363–398.

[15] Hans Meinhardt and Alfred Gierer. "Pattern formation by local self-activation and lateral inhibition". In: *Bioessays* 22.8 (2000), pp. 753–760.

[16] John L Nazareth. "Conjugate gradient method". In: *Wiley Interdisciplinary Reviews: Computational Statistics* 1.3 (2009), pp. 348–353.

[17] Hans G Othmer et al. "The intersection of theory and application in elucidating pattern formation in developmental biology". In: *Mathematical modelling of natural phenomena* 4.4 (2009), pp. 3–82.

[18] Chia-Ven Pao. “On nonlinear reaction-diffusion systems”. In: *Journal of Mathematical Analysis and Applications* 87.1 (1982), pp. 165–198.

[19] John Rinzel and David Terman. “Propagation phenomena in a bistable reaction-diffusion system”. In: *SIAM Journal on Applied Mathematics* 42.5 (1982), pp. 1111–1137.

[20] Hedi Sellami et al. “Accelerating the finite-element method for reaction-diffusion simulations on GPUs with CUDA”. In: *Micromachines* 11.9 (2020), p. 881.

[21] Kathrin Spendier and VM Kenkre. “Analytic solutions for some reaction-diffusion scenarios”. In: *The Journal of Physical Chemistry B* 117.49 (2013), pp. 15639–15650.

[22] István Szalai and Patrick De Kepper. “Turing patterns, spatial bistability, and front instabilities in a reaction- diffusion system”. In: *The Journal of Physical Chemistry A* 108.25 (2004), pp. 5315–5321.

[23] Alan Mathison Turing. “The chemical basis of morphogenesis”. In: *Bulletin of mathematical biology* 52 (1990), pp. 153–197.

[24] Sean T Vittadello et al. “Turing pattern design principles and their robustness”. In: *Philosophical Transactions of the Royal Society A* 379.2213 (2021), p. 20200272.

[25] Anatol M Zhabotinsky. “Belousov-zhabotinsky reaction”. In: *Scholarpedia* 2.9 (2007), p. 1435.

A The Stiffness and Damping Matrices

A.1 The Area of a Triangle

Given a triangle T with corners (x_i, y_i, z_i), (x_j, y_j, z_j), and (x_k, y_k, z_k), the area of the triangle A_T is

$$A_T = \frac{1}{2}\left|\begin{pmatrix} x_j - x_i \\ y_j - y_i \\ z_j - z_i \end{pmatrix} \times \begin{pmatrix} x_k - x_i \\ y_k - y_i \\ z_k - z_i \end{pmatrix}\right| = \frac{1}{2}\begin{vmatrix} (y_j - y_i)(z_k - z_i) - (y_k - y_i)(z_j - z_i) \\ (x_k - x_i)(z_j - z_i) - (x_j - x_i)(z_k - z_i) \\ (x_j - x_i)(y_k - y_i) - (x_k - x_i)(y_j - y_i) \end{vmatrix}.$$

In the 2D case where $z_i = z_j = z_k$, this becomes[1]

$$A_T = \frac{1}{2}\left[(x_j - x_i)(y_k - y_i) - (x_k - x_i)(y_j - y_i)\right] = \frac{1}{2}\left(x_i y_j + x_j y_k + x_k y_i - x_i y_k - x_k y_j - x_j y_i\right).$$

A.2 The Damping Matrix

The damping matrix $\mathbf{D}$ is defined so that

$$d_{i,j} = \int_\Omega \psi_i \psi_j dA.$$

I consider a triangle T with both v_i and v_j as vertices to T. From the solution for T, we know

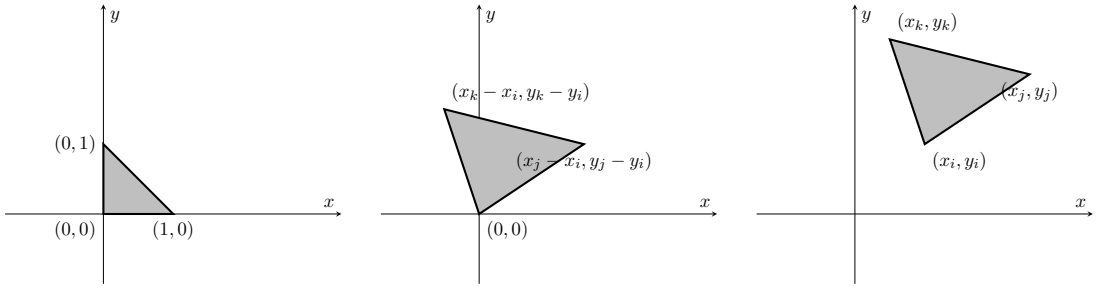
$$d_{i,j} = \sum_{\{T : v_i, v_j \in T\}} \int_T \psi_i \psi_j dA.$$

To calculate this integral over T, we will first define an affine transformation from T^0 to T where T^0 is a right triangle with $v_i = (0,0)$, $v_j = (1,0)$, and $v_k = (0,1)$ (Figure A.1). To map from $\tilde{x}, \tilde{y}$ on T^0 to T, we first scale and rotate according to the linear transformation

$$\begin{pmatrix} x_j - x_i & x_k - x_i \\ y_j - y_i & y_k - y_i \end{pmatrix} \begin{pmatrix} \tilde{x} \\ \tilde{y} \end{pmatrix}$$

[1] Technically, this could be the negative of the area and to get the actual area you need to take an absolute value. Solving it this way makes the next steps easier, however.

Figure A.1: Transformation from T^0 to T.



(a) Initial triangle T^0 (b) Rotated and scaled (c) Translated to T

then add $(x_i, y_i)^\top$ to get to T. Therefore, to get from $\tilde{x}, \tilde{y}$ on T^0 to x, y on T, we use

$$x = x_i + (x_j - x_i)\tilde{x} + (x_k - x_i)\tilde{y}$$
$$y = y_i + (y_j - y_i)\tilde{x} + (y_k - y_i)\tilde{y}.$$

The Jacobian of this transformation

$$\begin{pmatrix} x_j - x_i & x_k - x_i \\ y_j - y_i & y_k - y_i \end{pmatrix}$$

has determinant

$$\begin{vmatrix} x_j - x_i & x_k - x_i \\ y_j - y_i & y_k - y_i \end{vmatrix} = (x_j - x_i)(y_k - y_i) - (x_k - x_i)(y_j - y_i) = 2A_T.$$

Therefore, we get

$$\int_T \psi_i(x, y)\psi_j(x, y)dydx = 2A_t \int_{T^0} \psi_i(\tilde{x}, \tilde{y})\psi_j(\tilde{x}, \tilde{y})d\tilde{y}d\tilde{x}x.$$

On T^0, we have

$$\psi_i(\tilde{x}, \tilde{y}) = 1 - \tilde{x} - \tilde{y} \quad \text{and} \quad \psi_j(\tilde{x}, \tilde{y}) = \tilde{x}.$$

Therefore, we know

$$\begin{aligned}
\int_T \psi_i(x,y)\psi_i(x,y)dydx &= 2A_T \int_{T^0} \psi_i(\tilde{x},\tilde{y})\psi_i(\tilde{x},\tilde{y})d\tilde{y}d\tilde{x} \\
&= 2A_T \int_0^1 \int_0^{1-\tilde{x}} (1-\tilde{x}-\tilde{y})^2 d\tilde{y}d\tilde{x} \\
&= 2A_T \int_0^1 \int_0^{1-\tilde{x}} \left[1-2\tilde{x}-2\tilde{y}+2\tilde{x}\tilde{y}+\tilde{x}^2+\tilde{y}^2\right] d\tilde{y}d\tilde{x} \\
&= \frac{2}{3}A_T \int_0^1 (1-\tilde{x})^3 d\tilde{x} \\
&= \frac{2}{3}A_T \int_1^0 \hat{x}^3(-1)d\hat{x}, \quad \hat{x} = 1-\tilde{x} \\
&= \frac{1}{6}A_T
\end{aligned}$$

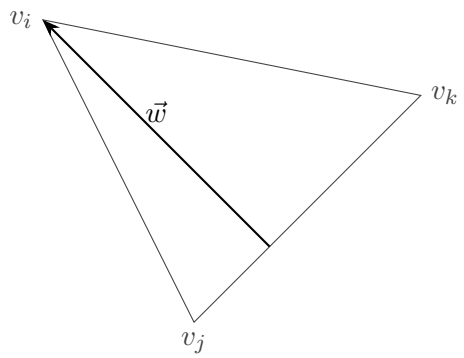
and

$$\begin{aligned}
\int_T \psi_I(x,y)\psi_j(x,y)dydx &= 2A_T \int_{T^0} \psi_i(\tilde{x},\tilde{y})\psi_j(\tilde{x},\tilde{y})d\tilde{y}d\tilde{x} \\
&= 2A_T \int_0^1 \int_0^{1-\tilde{x}} (1-\tilde{x}-\tilde{y})\tilde{x}d\tilde{y}d\tilde{x} \\
&= 2A_T \int_0^1 \tilde{x} \int_0^{1-\tilde{x}} (1-\tilde{x}-\tilde{y})d\tilde{y}d\tilde{x} \\
&= A_T \int_0^1 \tilde{x}(1-\tilde{x})^2 d\tilde{x} \\
&= A_T \int_1^0 (\hat{x}^2-\hat{x}^3)(-1)d\hat{x}, \quad \hat{x} = 1-\tilde{x} \\
&= \frac{1}{12}A_T.
\end{aligned}$$

Along a 3D surface, the same equation holds. We use the transformation

$$\begin{aligned}
x &= x_i + (x_j - x_i)\tilde{x} + (x_k - x_i)\tilde{y} \\
y &= y_i + (y_j - y_i)\tilde{x} + (y_k - y_i)\tilde{y} \\
z &= z_i + (z_j - z_i)\tilde{x} + (z_k - z_i)\tilde{y}
\end{aligned}$$

Figure A.2: $\vec{w}$ on a triangle



and the fact that the change in area elements is the ratio of the areas

$$\frac{A_T}{A_{T^0}} = \frac{A_T}{\frac{1}{2}} = 2A_T$$

to get to the same result.

A.3 The Stiffness Matrix

The stiffness matrix **S** is defined so that

$$s_{i,j} = \int_\Omega \nabla\psi_i \cdot \nabla\psi_j dA.$$

Like with the damping matrix, I will calculate the integral over a single triangle T. Then, we can calculate

$$s_{i,j} = \sum_{\{T:v_i,v_j\in T\}} \int_T \nabla\psi_i \cdot \nabla\psi_j dA.$$

To calculate the integral over T, recognize $\nabla\psi_i$ is constant over T. Therefore, we know

$$\int_T \nabla\psi_i \cdot \nabla\psi_j dA = A_T \nabla\psi_i \cdot \nabla\psi_j.$$

We know $\nabla\psi_i$ is a vector pointing into v_i orthogonal to $v_j - v_k$. Defining $\vec{w}$ to be orthogonal to $v_j - v_k$ such that

$$v_i - v_j = \lambda(v_j - v_k) + \vec{w}$$

for some scalar λ, we know

$$\vec{w} = v_i - v_j - \frac{(v_i - v_j) \cdot (v_j - v_k)}{(v_j - v_k) \cdot (v_j - v_k)}(v_j - v_k).$$

An example of this is given in Figure A.2. Then, we know $\frac{1}{|\vec{w}|}$ is the magnitude of the gradient[2] and $\frac{1}{|\vec{w}|}\vec{w}$ is the direction of the gradient. Therefore, the gradient is

$$\nabla\psi_i = \frac{1}{|\vec{w}|^2}\vec{w}.$$

In the 3D surface case, I directly implement this formula to calculate the stiffness matrix. In the 2D case, the $\vec{w}$ vector is

$$\begin{aligned}
\vec{w} &= \begin{pmatrix} x_i - x_j \\ y_i - y_j \end{pmatrix} - \frac{(x_i - x_j)(x_j - x_k) + (y_i - y_j)(y_j - y_k)}{(x_j - x_k)^2 + (y_j - y_k)^2} \begin{pmatrix} x_j - x_k \\ y_j - y_k \end{pmatrix} \\
&= \frac{1}{(x_j - x_k)^2 + (y_j - y_k)^2} \begin{pmatrix} (x_i - x_j)(y_j - y_k)^2 - (x_j - x_k)(y_i - y_j)(y_j - y_k) \\ (x_j - x_k)^2(y_i - y_j) - (x_i - x_j)(x_j - x_k)(y_j - y_k) \end{pmatrix} \\
&= \frac{(x_i - x_j)(y_j - y_k) - (x_j - x_k)(y_i - y_j)}{(x_j - x_k)^2 + (y_j - y_k)^2} \begin{pmatrix} y_j - y_k \\ x_k - x_j \end{pmatrix} \\
&= \frac{x_i y_j + x_j y_k + x_k y_i - x_i y_k - x_k y_j - x_j y_i}{(x_j - x_k)^2 + (y_j - y_k)^2} \begin{pmatrix} y_j - y_k \\ x_k - x_j \end{pmatrix} \\
&= \frac{2A_T}{(x_j - x_k)^2 + (y_j - y_k)^2} \begin{pmatrix} y_j - y_k \\ x_k - x_j. \end{pmatrix}
\end{aligned}$$

[2] Shoutout $\frac{\text{rise}}{\text{run}}$.

Then, the gradient is

$$\begin{aligned}\nabla\psi_i &= \frac{1}{|\vec{w}|^2}\vec{w} \\ &= \frac{(x_j - x_k)^2 + (y_j - y_k)^2}{2A_T}\left(\begin{pmatrix} y_j - y_k \\ x_k - x_j \end{pmatrix} \cdot \begin{pmatrix} y_j - y_k \\ x_k - x_j \end{pmatrix}\right)^{-1} \begin{pmatrix} y_j - y_k \\ x_k - x_j \end{pmatrix} \\ &= \frac{1}{2A_T}\begin{pmatrix} y_j - y_k \\ x_k - x_j \end{pmatrix}.\end{aligned}$$

This means the integrals over the gradients become

$$\int_T \nabla\psi_i \cdot \nabla\psi_i dA = \frac{1}{4A_T}\left((y_j - y_k)^2 + (x_k - x_j)^2\right)$$

and

$$\int_T \nabla\psi_i \cdot \nabla\psi_i dA = \frac{1}{4A_T}\left((y_j - y_k)(y_k - y_i) + (x_k - x_j)(x_i - x_k)\right)$$

which I plug into the stiffness matrix.