A Finite Element Approach to Reaction-Diffusion Systems

Gavin Engelstad*

gengelst@macalester.edu

Fall 2024

Abstract

Reaction-diffusion systems are a common framework to model important, real phenomena. Most methods to solve such systems are limited in the shape of the region they can solve the equation on. This paper uses an implementation of the finite element method to numerically solve reaction-diffusion systems on any domain. I test the method against a baseline example of a system on domains made up of 2D shapes and 3D surfaces. I also analyze the numerical properties of the system and solution method, finding that the system develops Turing patterns that vary across parameterizations and the solution method is very numerically stable, especially along smoother parts of the solution.

1 Introduction

Reaction-diffusion systems model the spatiotemporal behavior of chemical and physical systems of one of more components that react with each other and diffuse across space. Chemical reactions [30, 9], the human nervous system [6], population dynamics [2], and the patterns that show up on animal's skins [28, 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 [16].

*Thanks to Ashlyn Ryan for her chemistry help on this project. Replication code available at `https://github.com/GavinEngelstad/Reaction-Diffusion-FEM`.

The solutions to reaction-diffusion systems presents interesting patterns (dubbed "Turing Patterns" [29]), moving fronts, and oscillations [26, 22, 28].

Because of the wide range of applications for reaction-diffusion systems, 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 [21, 17]. This term drives the asymptotic behavior and stability of the system [21], but it also introduces additional complexity, especially for methods that abuse local, linear approximations of the system to solve it. Finite differences [11], spectral [1, 3], and analytic [24] approaches can solve this problem with various degrees of 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 txhis paper is heavily based on the mathematical approach of [23] and [14]. I utilize finite elements to discretize the space derivative and a combined implicit (backwards) and explicit (forwards) 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 [17]. I solve the system on the domain Ω and 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 [8]. The weak form is given by multiplying by some function v and integrating across the whole domain, which gets

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

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

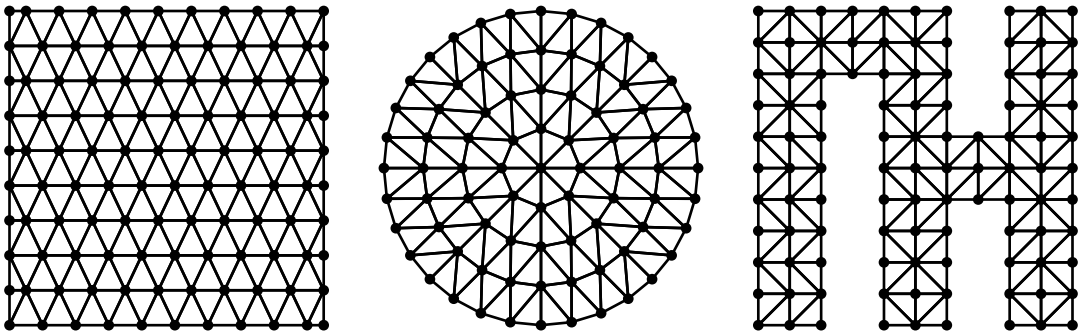
To discretize the spacial derivative, I triangulate Ω. Figure 2.1 gives examples of what potential triangulations look like for a variety of domains. Panel 2.1a shows this discretization for 2D domains including a square, disk, and maze-like structure and Panel 2.1b shows this discretization along the surface of 3D objects, including a sphere and a torus.

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

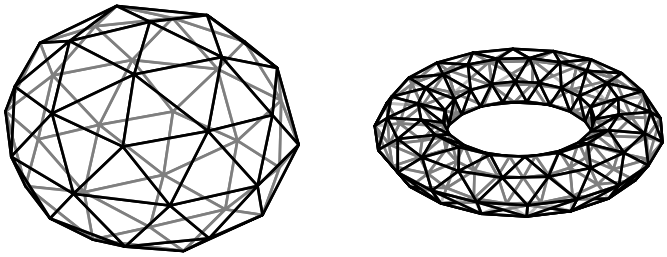
$$\psi_i(v_i) = 1$$
$$\psi_i(v_j) = 0, \quad j \neq i$$

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.

Figure 2.1: Triangulations of different domains



(a) 2D Domains



(b) 3D Surface Domains

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

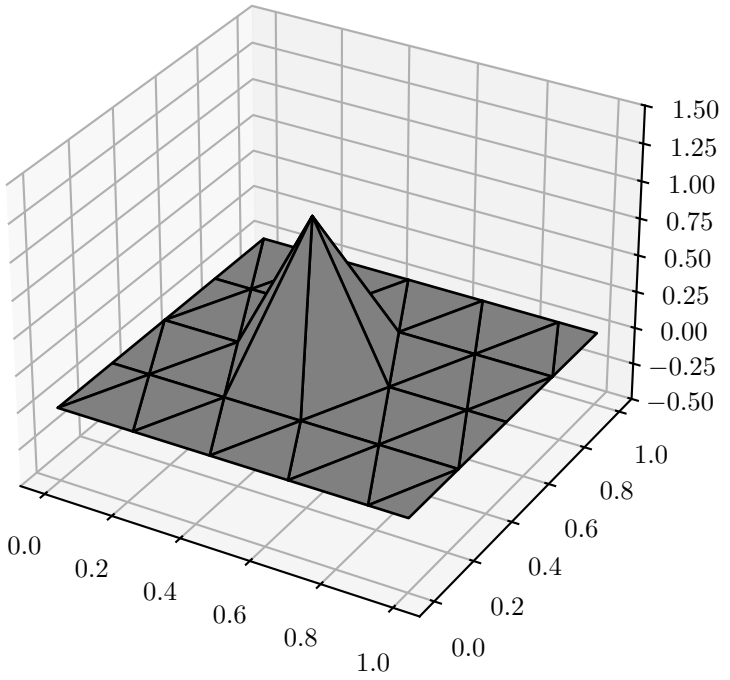
$$\hat{u}_n(t, x, y) = \sum_i \psi(x, y) u_{n,i}(t) \approx u(t, x, y).$$

Similarly, we can approximate r_n as

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

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 3.4 by solving systems with increasingly fine discretizations to get closer to the exact answer. Using these

Figure 2.2: The function ψ_i at a vertex



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 [23]. 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 [7]. 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

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

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 $\mathbf{u}_n^t$ [19]. 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, although any initial condition could be used. 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 [20, 12]. By replacing the reaction function, the method can be extended to any sufficiently smooth reaction-diffusion system. This includes Turing's initial version [28, 4] and other activator-inhibitor equations [13, 18] which it has been tested against. The system I solve is given by

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

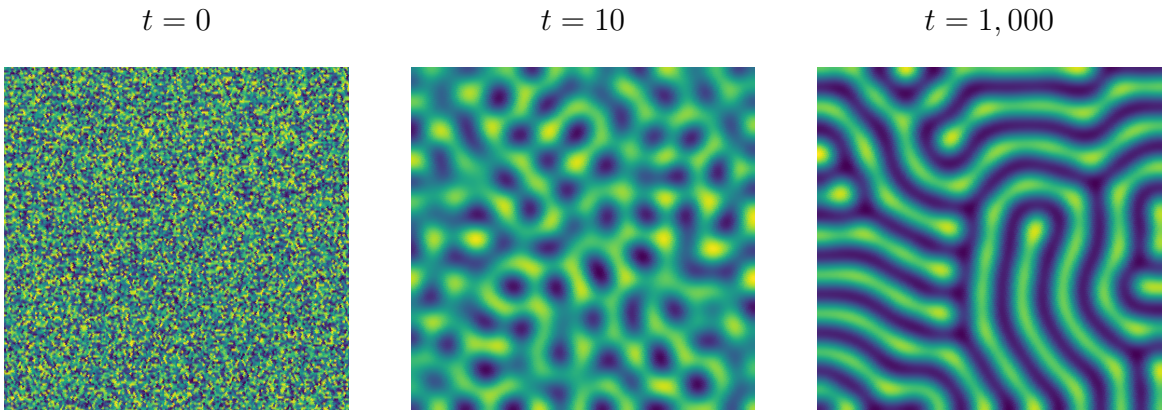
Like [12], I fix $D_u = 1$ and $k_2 = 11$ in all simulations. I vary D_v and k_1 across simulations to create different Turning patterns, but we always have $D_v < D_u$. The time step is set to $\Delta t = 0.01$, and all systems are iterated until $t = 1,000$, at which point they all have reached a steady 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

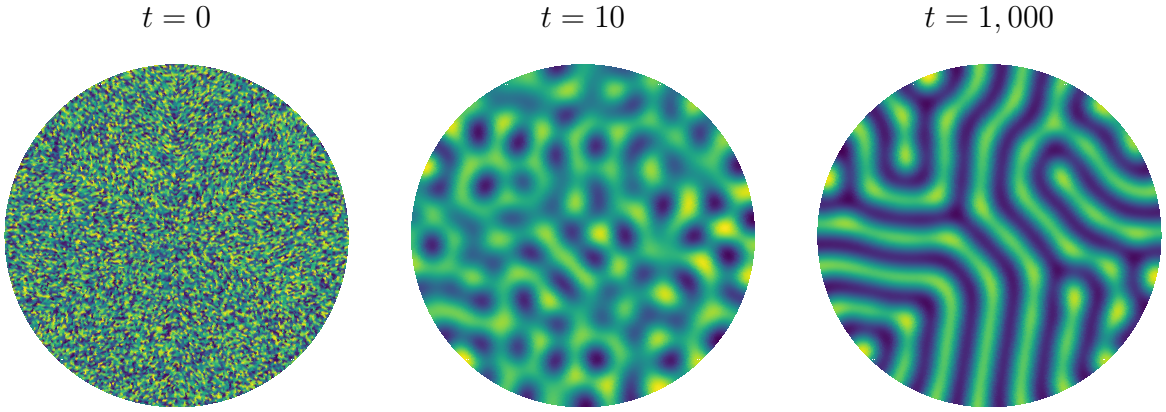
Figure 3.1 plots the solution to the reaction-diffusion system at different time steps on a square and circular domain. The left panel plots the initial condition, the middle panel plots an intermediate step during the reaction, and the right panel plots the system at steady state. I set the initial 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.

Figure 3.1: Solutions to the reaction-diffusion system



(a) 10×10 square domain



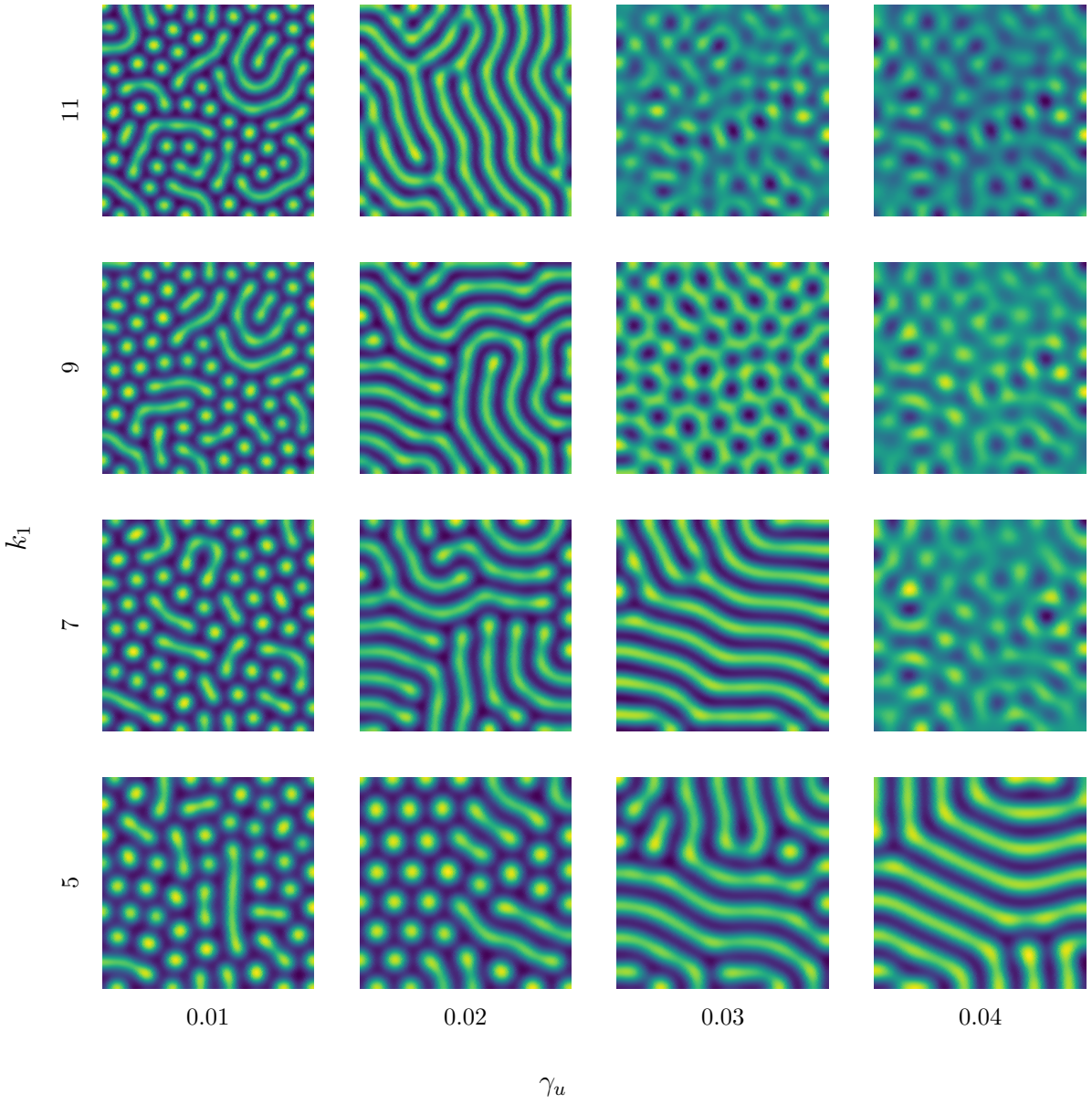
(b) Radius 5 circular domain

3.3 Parameters

The chosen parameterization has significant effects on the shape of the Turing patterns that develop. Specifically, alternate choices for the diffusion coefficient for v and the feed concentration k_1 can change significant features of the patterns.

Figure 3.2 shows how parameter choice affects the Turing patterns in $\gamma_v - k_1$ space on a square domain. With higher diffusion, the patterns are less defined having more fuzziness along the boundary and are wider. With low γ_v, spots emerge since the diffusion is not strong enough to merge the disjoint elements together. Increasing k_1 merges the spots together again forming stripes,

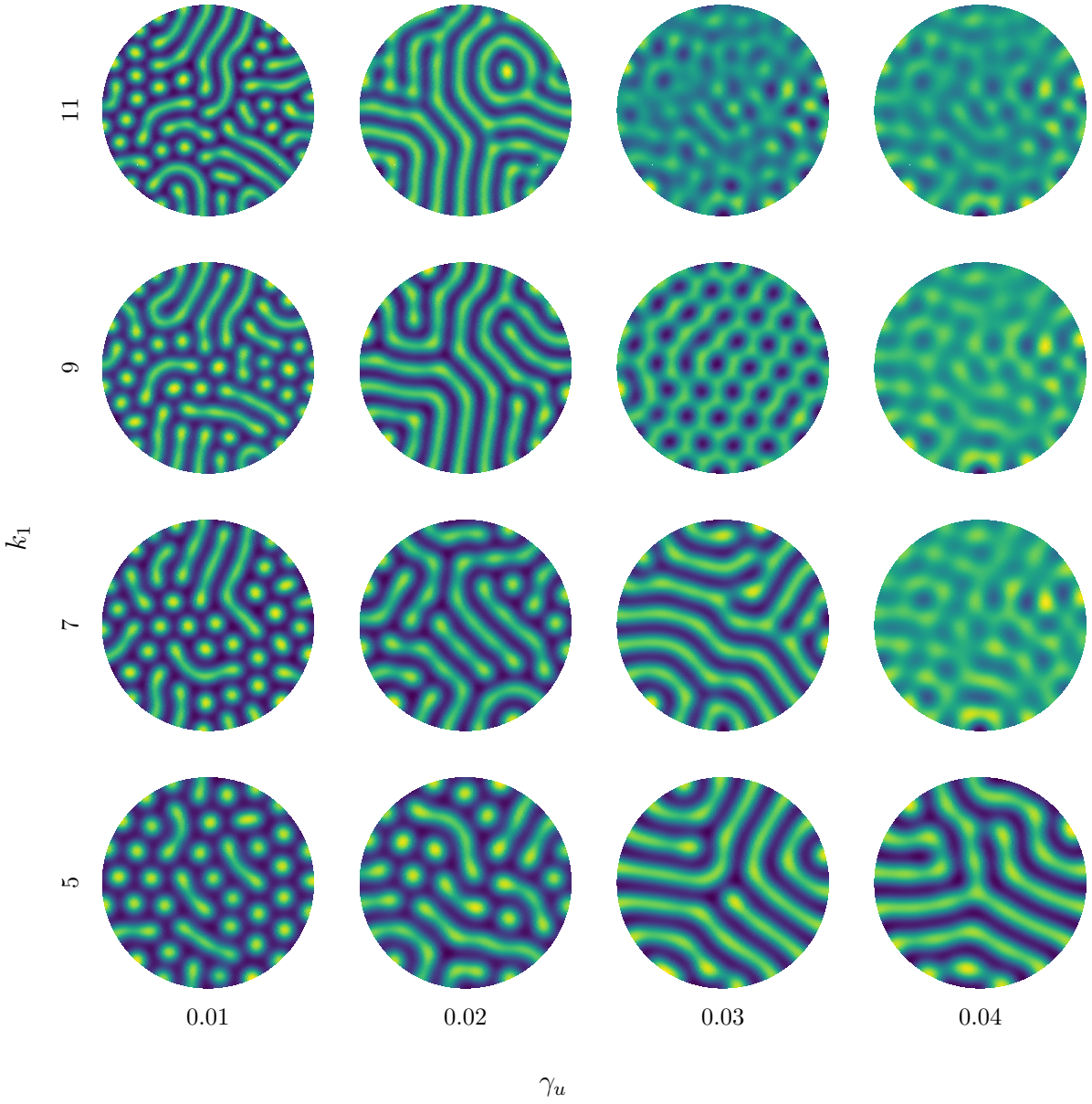
Figure 3.2: 10×10 square steady states for different γ_v and k_1



but does not have the same effect on the width and definition of the patterns. With high γ_v and k_1 the patterns become dominated by high-concentrations of u (yellow) instead of low concentrations (purple).

The parameterization has the same effect on alternate domains. Figure 3.3 shows the same parameterizations on a circular domain. Higher γ_v similarly increases the width and decreases the

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



definition of the Turing patterns, and higher k_1 merges the patterns together to form lines.

3.4 Error Analysis

By discretizing the grid, I am introducing potential numerical error into the solution. I analyze this by comparing solutions on less granular triangulations to the solution on a very fine triangulation.

Denoting $\hat{u}^N$ the finite element approximation for u on a grid with an average of N vertices along a 1 unit path across edges in the triangulation, I calculate

$$\epsilon_N = \int_\Omega \left| \hat{u}^N(\bar{t}, x, y) - \hat{u}^{\overline{N}}(\bar{t}, x, y) \right| dA$$

for some large $\bar{t}$ so that the system has reached the steady state and large $\overline{N}$ to get as accurate an approximation as possible of the exact solution. I use $\overline{N} = 50$, which means each square unit in the domain has approximately 2,500 vertices in it. I evaluate $\hat{u}^N$ with $k_1 = 9$ and $\gamma_v = 0.02$ on an evenly spaced 100×100 grid then use a Riemann sum to calculate the integral. For the initial condition, I use OpenSimplex noise [27]. This ensures all N have the same u at $t = 0$.

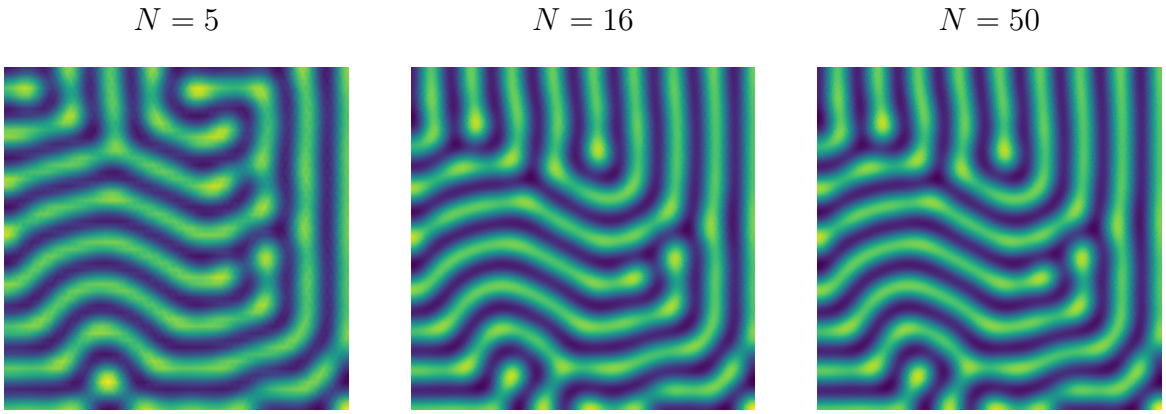
Figure 3.4 gives the results of this analysis. Panel 3.4a shows the resulting steady states using different N. Qualitatively, the Turing patterns that develop look very similar across N. With low N, the lines do form different connections, especially in the top right corner where the lines form together differently when $N = 16$ and $N = 50$ versus when $N = 5$. The patterns in the $N = 16$ and $N = 50$ case are visually identical.

Panel 3.4b shows the total error for different values of N and the comparison between the $N = 16$ and $N = 50$ case. In general, a larger value of N reduces error, although there is diminishing returns since the magnitude of the error decrease is lower at higher N. For the $N = 16$ case, total error is low across the whole domain. The error follows wave-like patterns through the domain. Comparing the locations of the error with the patterns in Panel 3.4a, the points with the highest error are located on the boundaries of the Turing patterns. This is consistent with the fact that the FEM approximation is worse in areas where the function is less smooth.

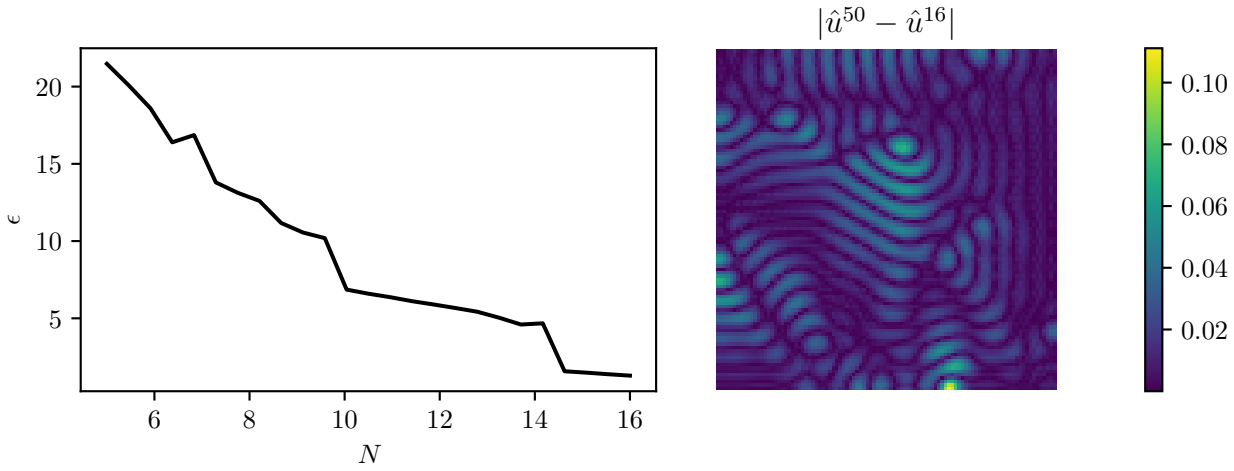
4 More Complex Domains

The main advantage of the FEM over finite differences or spectral methods is the ability to handle oddly shaped domains [5]. This section explores solutions to reaction-diffusion equations on oddly shaped domains that would be hard or even impossible to solve using other approaches.

Figure 3.4: Numerical error in the solutions on a 10×10 square domain



(a) Calculated steady state with different N



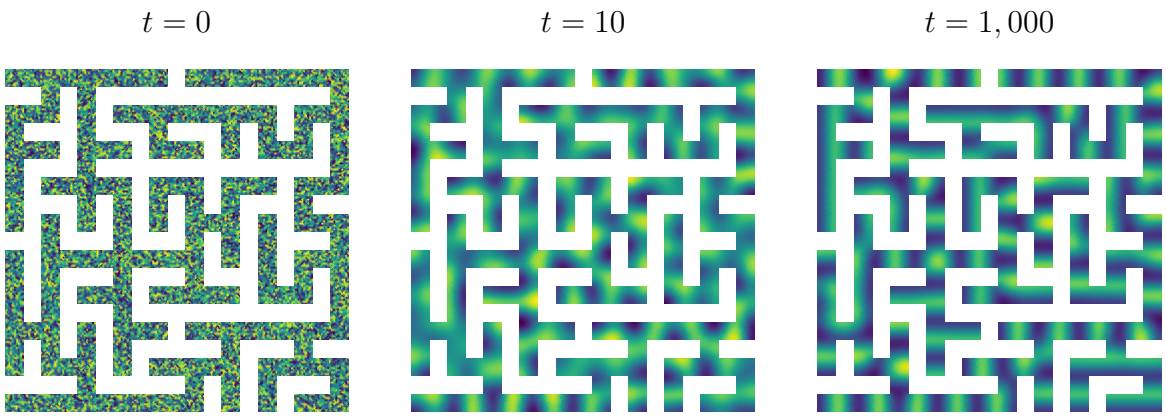
(b) (Left) Error by N; (Right) Error over Ω for $N = 16$

4.1 A 2D Maze

In 2D space, I show this using a domain with a maze-like shape. By defining a triangulation within the space (Figure 2.1), the finite element approach can solve the reaction-diffusion system within this abnormal shape.

Figure 4.1 shows the solution to the PDE with $k_1 = 9$ and $\gamma_v = 0.02$ on this domain. Starting from random noise at $t = 0$, the solution starts to form structure by $t = 10$, and form Turing patterns at $t = 1,000$ in the steady state. Interestingly, the lines within the Turing patterns are

Figure 4.1: Solution to the reaction-diffusion system on a 10×10 maze-like domain



all either parallel or perpendicular to the boundary. Because the spacing between lines is relatively consistent, the patterns roughly line up with each other. They do slightly misalign, however, especially on the right side, demonstrating the effect of the domain choice on the solution.

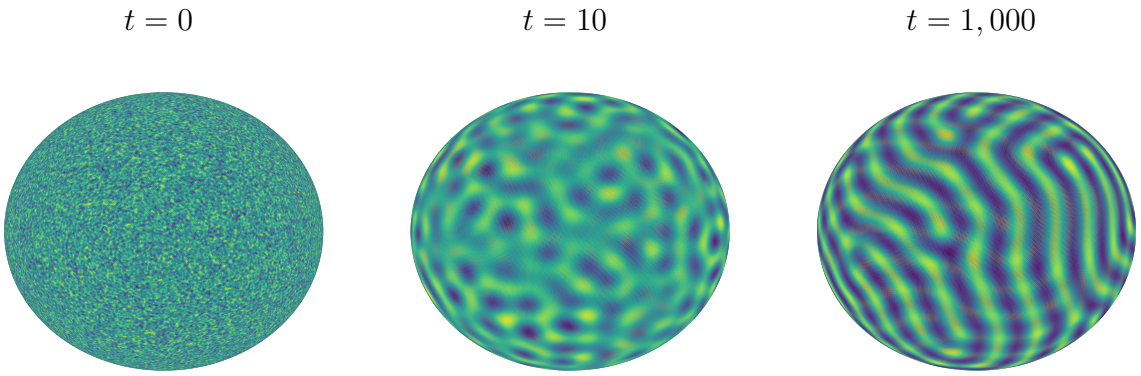
4.2 3D Surfaces

Because the surface of 3D shapes can also be triangulated, I can also use the FEM to solve the system along 3D surfaces. My approach ignores the effect of the curvature of the surface, which can affect the shape of the patterns that emerge [25, 15]. Still, it gives a good approximation for the solution of the system on the surface.

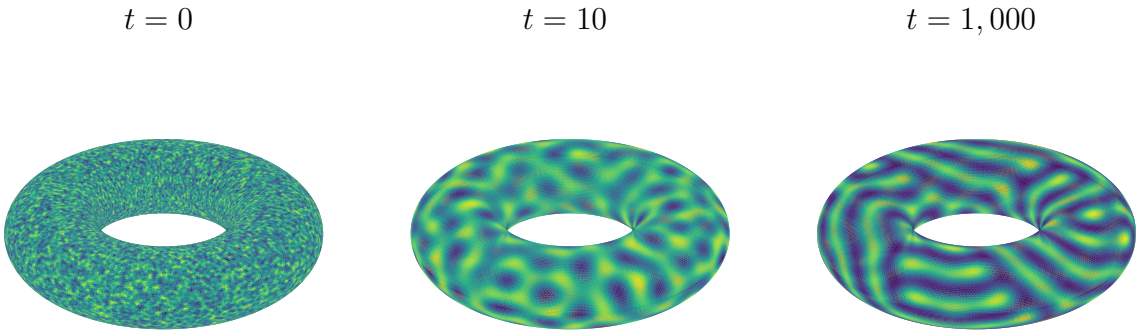
Figure 4.2 shows the solution to the system with $k_1 = 9$ and $\gamma_v = 0.02$ on two 3D surfaces. Panel 4.2a shows the solution along a sphere and Panel 4.2b shows the solutions along a torus. Each point in the triangulation for both shapes is connected to its neighbors with no seam, unlike [15], so we allow for diffusion across the whole domain.

The solutions on both surfaces follow the same general pattern as the solution on 2D domains in Figures 3.1 and 4.1. At $t = 0$, the domain is covered in random noise. Then, at $t = 10$, we see basic structure emerging before clear Turing patterns appear at $t = 1,000$ in the steady state. On the torus, the lines in the Turing pattern are almost completely parallel to each other while on the sphere more complex patterns develop.

Figure 4.2: Solution to the reaction-diffusion system on 3D surfaces



(a) Radius 5 sphere domain



(b) $R = 3.5$, $r = 1.5$ torus domain

5 Conclusion

This paper presets a finite element approach to solving reaction-diffusion PDEs. I discretize the spacial derivative for the diffusion term according to the finite element method. Then, I combine the implicit and explicit Euler methods to calculate numerically stable time steps given the nonlinear reaction. Because I use the FEM, I can solve reaction-diffusion systems on a wide range of domains, including ones with complex structure and on 3D surfaces.

Across all surfaces, the reaction-diffusion equation develops similarly from random noise, to diffuse hotspots, and finally to well-defined Turing patterns. The Turing patterns that develop

depend on the parameter choices within the PDE, but the general qualitative properties, including the shape and width, of the patterns are robust across square and circle domains. Exploration into the numerical properties of the method suggest it generally gets a reasonable degree of accuracy, but can have some error, especially on the less smooth parts of the function.

To expand on this method, future projects should search for more accurate solutions with more realistic applications. One approach would be to implement the spectral element method [10] and use polynomials in place of my linear ψ_i functions. Alternatively, on 3D domains the curvature of the domain can affect the solution to the PDE [25, 15], so future work could look into the effect of this curvature. Specific to animal pigmentation patterns, a more realistic model would also incorporate the effect of tissue growth in reaction-diffusion systems [4].

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.

[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.

[5] ID Erhunmwun and UB Ikponmwosa. "Review on finite element method". In: *Journal of Applied Sciences and Environmental Management* 21.5 (2017), pp. 999–1002.

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

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

[8] 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`.

[9] 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.

[10] Muhammad Bilal Hafeez and Marek Krawczuk. "A review: Applications of the spectral finite element method". In: *Archives of Computational Methods in Engineering* 30.5 (2023), pp. 3453–3465.

[11] 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.

[12] 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.

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

[14] 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.

[15] D Assaely León-Velasco and Guillermo Chacón-Acosta. “Full Finite Element Scheme for Reaction-Diffusion Systems on Embedded Curved Surfaces in $\mathbb{R}^3$”. In: *Advances in Mathematical Physics* 2021.1 (2021), p. 8898484.

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

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

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

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

[20] 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.

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

[22] 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.

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

[24] 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.

[25] Michael F Staddon. “How the zebra got its stripes: Curvature-dependent diffusion orients Turing patterns on three-dimensional surfaces”. In: *Physical Review E* 110.3 (2024), p. 034402.

[26] 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.

[27] Yann Thorimbert and Bastien Chopard. “Polynomial methods for fast procedural terrain generation”. In: *arXiv preprint arXiv:1610.03525* (2016).

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

[29] 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.

[30] 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

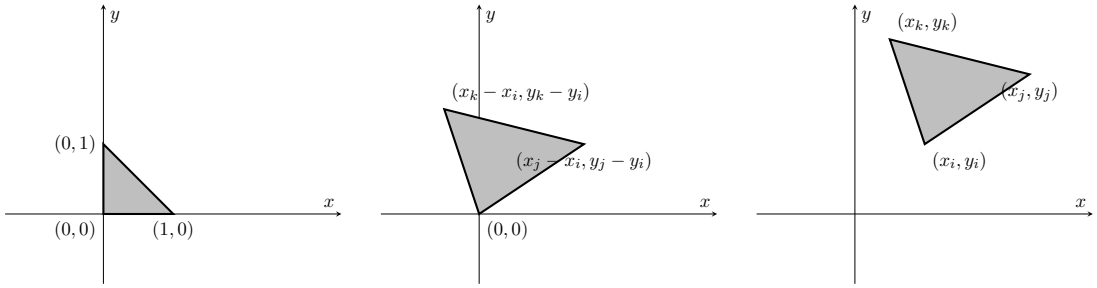
$$d_{i,j} = \sum_{\{T : v_i, v_j \in T\}} \int_T \psi_i \psi_j dA.$$

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

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

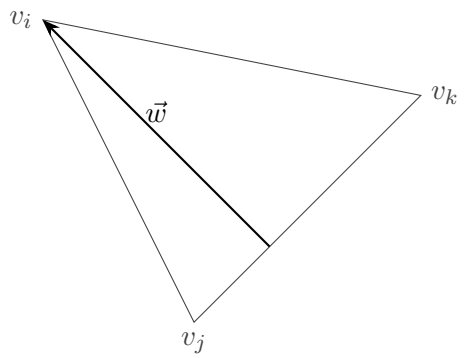
and

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

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.