Discussing the Use of Reaction Diffusion Equations for the Simulation of Physical Phenomena

Simon Abrelat

002129–0004

IB Extended Essay

February 2019

Words: 2550

Contents

1 Introduction **2**

2 Calculus **3**

3 Diffusion **10**

4 Reaction **11**

4.1 Logistic Equation 12

4.2 Lotka-Volterra Model 13

4.3 SIR Model 15

5 Reaction Diffusion **17**

5.1 Fisher's equation 17

5.2 Cahn-Hilliard 19

5.3 Gray-Scott 20

6 Conclusion **22**

7 Appendix **26**

1 Introduction

Reaction Diffusion equations are a class of equations with a wide range of uses and applications. They allow for the modeling of the iterations between objects throughout space while passing through time. In other words, they track the relationship between things over time as they spread and affect other things. This is not a in-depth definition, but it serves to understand their purpose. It should be evident from the generality of their purpose that their applications are equally universal. As the name implies, they have two sections: the reaction and the diffusion. The reaction serves as the relation between objects and the diffusion is the spacial component. This is a discussion to visually and intuitively explain the diffusion and reaction components and to tie them together to show the complex and emergent behavior of Reaction Diffusion systems.

The thing that the Reaction Diffusion takes and manipulates or the 'object' previously referred to is a concentration variable, and the goal is to morph a concentration space. The equation takes a field of values and returns a field or matrix of modified values. One

way to think about the Reaction Diffusion is that there is a reaction that spreads getting more 'fuel' to continue the reaction. Another examples of this sort of reaction is a disease that starts with one person and spreads based off of who touched it and stops when everyone has recovered or died. Given the parameters of the equation there are a variety of intuitive explanations with different levels of accuracy or mathematical formality.

2 Calculus

Reaction Diffusion relies on multivariable calculus, from the partial derivative to the Laplacian operator that governs diffusion. Multivariable calculus is a natural extension of calculus where instead of focusing on the accumulation or change on 1 axis, that normally being y, it is possible to run the calculus operations for any number of dimensions. This means that if a function has more than 1 input, for instance 3D functions, there is a whole new arsenal of tools and methods to analyze these equations. The limit fundamentally operates the same as in the 2 dimensional case. If there were a hole in the function

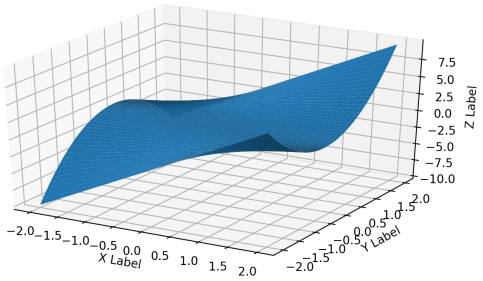
it would 'fill in' that hole, and if there is a jump discontinuity the limit is defined for certain directions as it approaches the discontinuity but the limit itself is undefined. Once again, integral and derivative can be derived from the limits as in normal calculus; however, this application will be primarily focusing on partial derivatives and other related operations. Instead derivatives in multivariate partial derivatives are used. These are computed in a similar form to derivatives but the inputs that are not differentiated is treated as a constant. For use in the reaction equations the partial differential equations behave in a similar way to differential equations except they have more inputs, but still give the rate of change for a given input over a function.

$$\frac{\partial f}{\partial x} = \partial_x f = \lim_{h \to 0} \frac{f(x+h, y) - f(x, y)}{h} \tag{1}$$

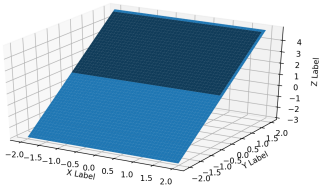
$$\frac{\partial f}{\partial y} = \partial_y f = \lim_{h \to 0} \frac{f(x, y+h) - f(x, y)}{h} \tag{2}$$

Example problem

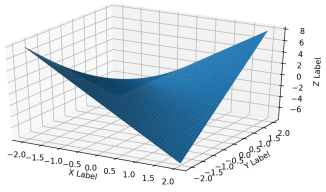
$$f(x, y) = xy^2 + x$$
$$\partial_x f(x, y) = y^2 + 1$$
$$\partial_y f(x, y) = 2xy$$



(a) $f(x, y) = xy^2 + x$



(b) $\partial_x f(x, y) = y^2 + 1$



(c) $\partial_y f(x, y) = 2xy$

Figure 1: Example functions

Directional derivatives are similar to the partial derivatives, but instead of being in a component direction, along the x or y direction, it would be in the direction of an arbitrary vector ,$\vec{u}$, with components $< a, b >$. These derivatives can be done in any number of dimensions (3) but for Reaction Diffusion formulas shown 2D cases (4) matter more.

$$D_u f(\vec{x}) = \lim_{h \to 0} \frac{f(\vec{x} + hu) - f(\vec{x})}{h} \tag{3}$$

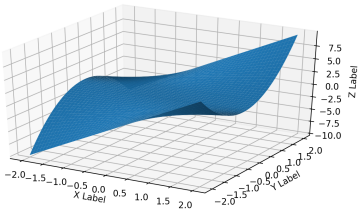
$$D_u f = \lim_{h \to 0} \frac{f(x + ha, y + hb) - f(x, y)}{h} \tag{4}$$

The directional derivative can also be written as a dot products.

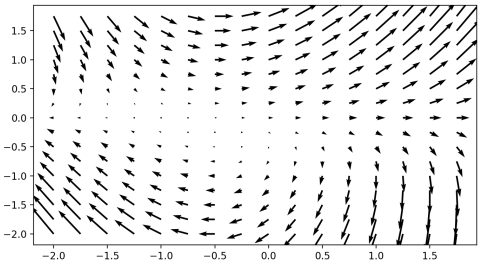
$$\begin{aligned} D_u f(x, y) &= a\partial_x f + b\partial_y f \\ &=< \partial_x f, \partial_y f > \cdot < a, b > \\ &=< \partial_x f, \partial_y f > \cdot u \\ &= \nabla f(x, y) * u \end{aligned}$$

This ∇, or nabla, is the sign for the grad operator. This represents a vector of all of the partial derivatives for all of the function inputs, which is called the gradient. For example in $g(x, y, z)$, the gradient of g is the vector where each component is the partial derivative of g for its inputs x, y, and z. This gradient points in the direction of the 'fastest increase' of the function. Picture a map of a mountain range the gradient of the map from a point would be the fastest way to get to the top of the nearest mountain but not necessarily the highest mountain. The gradient is often used for optimization problems since it can go 'the highest' or local maxima in a function of any number of variables. A common abuse of notation is to set ∇ to a value (5)

$$\begin{aligned}\nabla &=< \frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z} > \\ &= \sum_{i=1}^{n} \vec{e_i} \frac{\partial}{\partial x_i}\end{aligned} \tag{5}$$



(a) $f(x, y) = xy^2 + x$



(b) the gradient vector field: $\nabla * f(x, y)$

The notation for the gradient makes sense the function, f, would be treated as a scalar and multiplying it by the ∇ vector. This abuse of notation since it would also make sense for the divergence of a vector field. For a vector field the function $\mathbf{F}$ (6) would be a vector function of the dot product of $\mathbf{F}$ and ∇ (7)

$$\mathbf{F} = P\hat{\imath} + Q\hat{\jmath} + R\hat{k} \tag{6}$$

$$div\, \mathbf{F} = \nabla * \mathbf{F} = \frac{\partial P}{\partial x} + \frac{\partial Q}{\partial y} + \frac{\partial R}{\partial z} \tag{7}$$

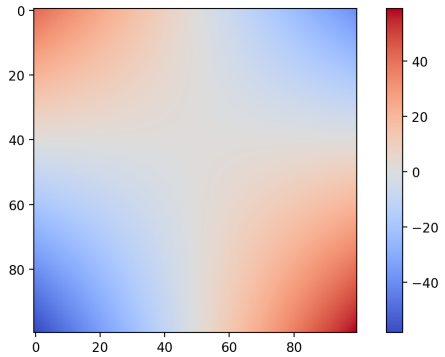


Figure 3: the gradient vector field: $\nabla * f(x, y)$

The divergence being the dot product of the partial derivative vector makes sense since it is the sum of the derivatives or 'velocities'. One can picture a vector field as being a model of fluid flow, where the vector at a point is velocity of the fluid. Given this analogy, the divergence is the rate that the fluid 'speeds up' at a point. The divergence if positive when fluid 'speeds up' and negative if it 'slows down'. In a vector field represented by all of the arrows pointing away from the origin and the arrows get larger around a point when the divergence $>$ 0 and if the arrows are smaller or point to the origin when the divergence < 0. Both of these tools are useful but when applied together become what is needed for the Diffusion part of Reaction Diffusion, that being the laplacian. The laplacian (8) is the divergence of the

gradient of a function.

$$\begin{aligned} \nabla^2 f &= \nabla * (\nabla f) \\ &= \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} + \frac{\partial^2 f}{\partial z^2} \\ &= \sum_{i=1}^{n} \frac{\partial^2 f}{\partial x_i^2} \end{aligned} \tag{8}$$

Equation 8 is called the Laplace operator, or the Laplacian of a field. The reason that,when put in partial differiental equations (PDEs), they exhibit this diffusion or spreading property, is due to their relation to the second derivative in normal calculus. They identify local maxima and minima, but in normal calculus where the second derivative is 0, in both cases the Laplacian is highly positive in minima and highly negative in maxima. The Laplacian is the sum of second partial derivatives for all of the function's inputs. In the case of PDEs, the laplacian 'fills in' the minima and 'pushes down' the maxima. This process is what drives osmosis and heat diffusion.

3 Diffusion

Diffusion is the spread of things from high to low densities. A simple example would be diffusion with heat,or osmosis with water. The mathematical way this is described is through a partial differential equation using the Laplace operator, ∇^2.

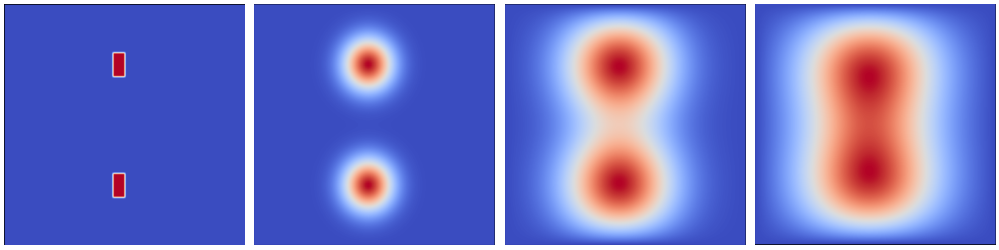
$$\frac{\partial u}{\partial t} = \alpha \nabla^2 u \tag{9}$$

$$\frac{\partial u}{\partial t} = \alpha(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}) \tag{10}$$

Equation 9 is a simple partial differential equation that has $u(x, y, z)$ to determine the heat and α as the coefficient of thermal diffusivity. Because of the Laplace operator, the equation does not need to show its inputs which also allows for a n-dimensional continuation. In 3 dimensions, this equation can be expanded into equation 10.

Figure 4 shows the diffusion from 2 point sources of heat that eventually merge into a single blob that would continue to spread throughout the plane.

Figure 4: Heat Diffusion Progression



(a) $n = 0$ (b) $n = 1000$ (c) $n = 5000$ (d) $n = 8000$

The heat diffusion equation (9) uses the Laplacian which 'pushes' the local maxima down and the local minima up. Later in the Reaction Diffusion equations, the Laplacian is used to give the equations properties based off their location and physical proximity to other elements. This diffusion section gives reaction diffusion equations their interesting properties since previously converging reactionary systems remain in this possibly unstable equilibrium.

4 Reaction

Reaction systems that typically contain multiple ordinary or partial differential equations. Where diffusion equations usually have only one variable, reaction systems are often multi-variable. There are many different examples of reaction systems, and this section is going to focus

on the logistic equation and SIR and Lotka-Volterra models.

4.1 Logistic Equation

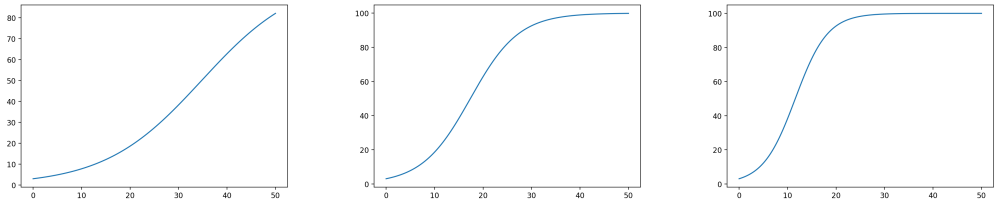
The most simple and generally recognized ordinary differential equation reaction system, especially relating to population, would be the logistic equation (11) which is a 1 component system that shows the growth and maximum population of an area.

$$\frac{dP}{dt} = rP(1 - \frac{P}{K}) \tag{11}$$

r: growth rate

K: carrying capacity

Figure 5: Logistic equation $p_0 = 3$ $k = 100$



(a) $r = 0.1$ (b) $r = 0.2$ (c) $r = 0.3$

The equation that describes the S-curve of growth and decay of the population. rP is the unencumbered growth rate of the population, which is proportional to the current population. $1 - \frac{P}{K}$ is the limiting

factor for the logistic equation. As your population approaches the carrying capacity the growth slows, so the point they are equal and there is no more growth. If the population increases over the carrying capacity it also pulls the population back into equilibrium This equation was originally generated for describing Malthusian population growth, and was later rediscovered a number of times. An examples of the use of the logistic equation is in modeling of tumor growth (Foryś and Marciniak-Czochra, 2003).

4.2 Lotka-Volterra Model

Another, and more complex, reaction system is the Lotka-Volterra model (12). This is used to model the populations of predators and pray as they grow and shrink in a cyclical manner, making it a 2 component system, in the case x for prey and y for predators. This is one of the earliest Predator-Pray models based on sound mathematical principles. There are some problems with such a simple model, but the biggest issue is, in reality populations, reach an equilibrium, but this model is cyclic. There are also cases were there is a cyclic

growth off to infinity, becoming unstable (Zhong et al., 1999). These issues mainly come from a lack of gender differences, size effects such as overpopulation and over consumption of resources, and a variety of outside influences. Hence why this is only used to describe specific predator-prey examples such as hare and lynx populations.

$$\begin{aligned} \frac{dx}{dt} &= \alpha x - \beta xy \\ \frac{dy}{dt} &= -\gamma y + \delta xy \end{aligned} \tag{12}$$

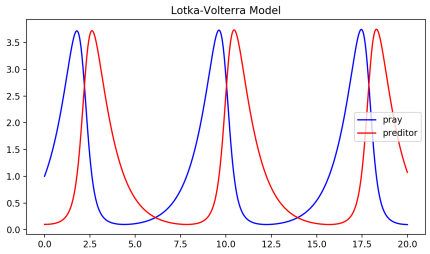
α: the intrinsic rate of pray population growth

β: predation rate coefficient

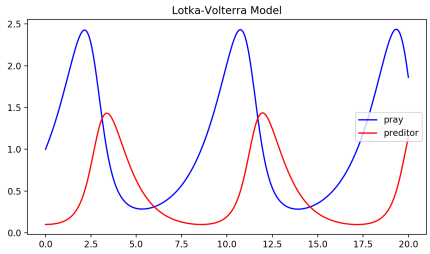
γ: the predator mortality rate.

δ: reproductive rate of the predators per pray eaten

Figure 6: Examples of the Lotka-Volterra Model



(a) $\alpha = \beta = \gamma = \delta = 1$



(b) $\alpha = \frac{2}{3}, \beta = \frac{4}{3}, \gamma = 1, \delta = 1$

The benefit of such a simple model is the simple explanation. For

prey, αx is clearly the unencumbered growth rate and $-\beta xy$ is the rate of predation that is influenced by the size of both the predator and prey populations. The predators explanation is a little more complicated. For predators, the δxy is the increase in predator population, given that each predator reproduces at a rate δ per prey eaten. The $-\gamma y$ is the mortality rate for the predators. This relates to Reaction Diffusion since they illustrates how multiple systems can be used in relation to each other before Diffusion.

4.3 SIR Model

The SIR model (13), or Susceptible-Infected-Recovered Model, is a computational epidemiology and serves as the core of other more advanced models, despite already having 3 components. This model is fairly intuitive. The susceptible equation is similar to the predation in the Lotka-Volterra Model, were the population decreases by the rate of spread times the infected and the susceptible, $-\beta si$. Therefore, if the size of the infected or susceptible populations are low, then there are few new cases of illness. The recovering is simply decreasing by the

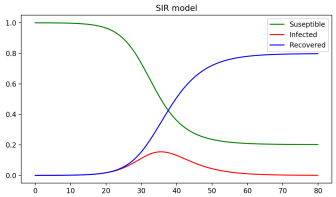
recovery rate times infected, κr. The infected change is equal to the newly infected minus the recovered, $\beta si - \kappa r$. Figure 7a is an example for a typical illness that is not an epidemic. In comparison, Figure 7b is an example of an epidemic.

$$\begin{aligned} \frac{dS}{dt} &= -\beta si \\ \frac{dI}{dt} &= \beta si - \kappa r \\ \frac{dR}{dt} &= \kappa r \end{aligned} \tag{13}$$

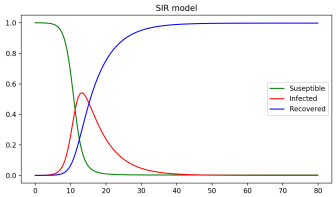
κ: the number of days until recovery

β: the number of days needed to be in contact to spread

Figure 7: Examples of SIR model



(a) $\kappa = \frac{1}{3}, \beta = \frac{1}{2}$



(b) $\kappa = \frac{1}{6}, \beta = 1$

5 Reaction Diffusion

Reaction Diffusion is the merger of diffusion and reaction systems with emergent properties. The generalized formula (14) works in any number of dimensions and with any number of components. To limit algorithmic complexity, only the 1^{st} and 2^{nd} dimensional cases will be discussed. $R(\vec{q})$ is a vector equation that represents the reaction systems described in Section 4. To complete the name, the $\underline{\underline{\mathbf{D}}}\nabla^2\vec{q}$ then represents the diffusion for any number of dimensions, exactly like heat diffusion described in Section 3.

$$\frac{\partial \vec{q}}{\partial t} = -\underline{\underline{\mathbf{D}}}\nabla^2\vec{q} + R(\vec{q}) \tag{14}$$

$\underline{\underline{\mathbf{D}}}$ is a diagonalized weight matrix

$\vec{q}$: a vector of inputs given a time

$R(\vec{q})$: a function that takes into account local reactions

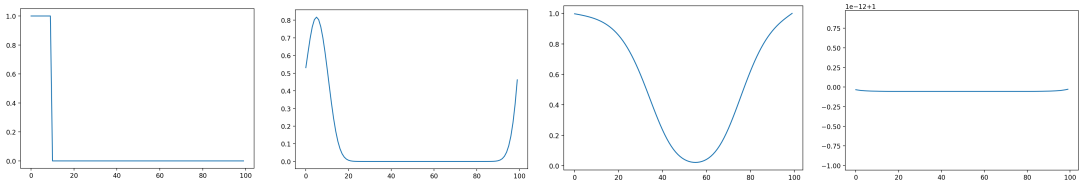
5.1 Fisher's equation

The first reaction diffusion system is Fisher's equation (15) which is a 1 component, 1 dimensional system that represents the wave front switching between equilibrium states, such as a switch of dominant

genetic traits (R.A.Fisher, 1937).

$$\frac{\partial u}{\partial t} - D\frac{\partial^2 u}{\partial x^2} = ur(1-u) \tag{15}$$

Figure 8: Fisher Equation Example



(a) $t = 0$ (b) $t = 50$ (c) $t = 500$ (d) $t = 10000$

The Fisher equation (15) is related to the logistic equation (11) previously described. The logistic equation can be thought of as the $R(\vec{q})$ given in the generalized reaction diffusion equation (14) and the $\frac{\partial^2 u}{\partial t^2}$ is the $-\underline{\underline{\mathbf{D}}}\nabla^2\vec{q}$ of the general form. The Fisher equation is not as generalized as the logistic equation, since the 'carrying capacity' is 1. These equations are still useful for illustrating the relation between reaction and reaction-diffusion systems.

5.2 Cahn-Hilliard

The Cahn-Hilliard equation (16) is a 1 component, n dimensional system related to phase separation (Cahn and Hilliard, 1958). Where Fisher's equation (15) is about the phase transition in 1 dimension, which is a wave, the Cahn-Hilliard equation can be used to describe the phase transition on a plane or any space, $\mathbb{R}^n$. This system can be applied to a multitude of uses, a theme of reaction diffusion equations.

$$\begin{aligned} \frac{\partial c}{\partial t} &= D\nabla^2(c^3 - c - \gamma\nabla^2 c) \\ &= D\nabla^2\mu \end{aligned} \tag{16}$$

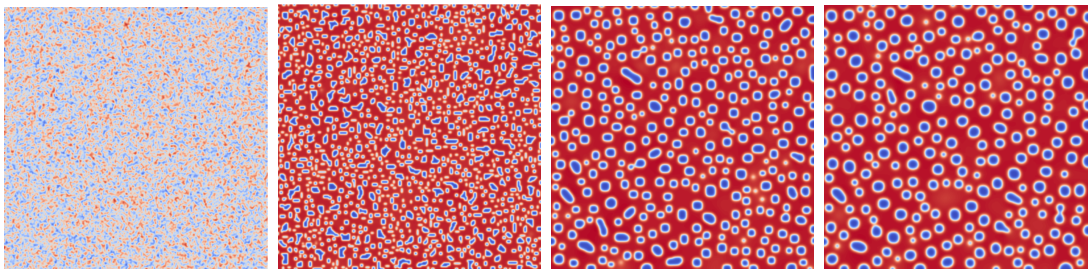
c: the concentration of the liquid

D: the diffusion coefficient

$\sqrt{\gamma}$: the length of the transition regions between the domain

$\mu = (c^3 - c - \gamma\nabla^2 c)$: the called the chemical potential

Figure 9: Cahn-Hilliard Examples at $t = 2000$



(a) Initial Seed (b) $\gamma = 0.125$ (c) $\gamma = 0.375$ (d) $\gamma = 0.5$

One such use is in high-strength alloys (F. J. Vermolen, 2009). An intuitive way of thinking about this process, despite losing some of the generality, is thinking about the Cahn-Hilliard equation as the interaction between insoluble fluids. If the γ, chemical potential, is high enough, as t approaches ∞, the two insoluble fluids completely separate. It is clear that the Cahn-Hilliard equation is very useful for studying emulsions and solutions as shown by Joon Choi and Anderson (2012) and other studies.

5.3 Gray-Scott

The Gray-Scott Model is the reaction diffusion equation for computational art. Originally, it was for modeling a specific chemical reaction (Gray and Scott, 1984). Due to its generality and interesting patterns it became used for making natural-looking patterns with a seed, for example the Gray-Scott lamp (Jessica, 2010). The original reaction takes in a and b then produces a byproduct, p. k is the rate that b turns into the nonreactive p. f is the rate that a is added and the other components are removed. The Gray-Scott model is also

used computational artists. Its main uses are for modeling patterns in biology.

$$\begin{aligned} \frac{\partial a}{\partial t} &= d_a \nabla^2 a - ab^2 + f(1-a) \\ \frac{\partial b}{\partial t} &= d_b \nabla^2 b + ab^2 - (k+f)b \end{aligned} \tag{17}$$

$$\begin{aligned} a + 3b &\rightarrow 3b \\ b &\rightarrow p \end{aligned} \tag{18}$$

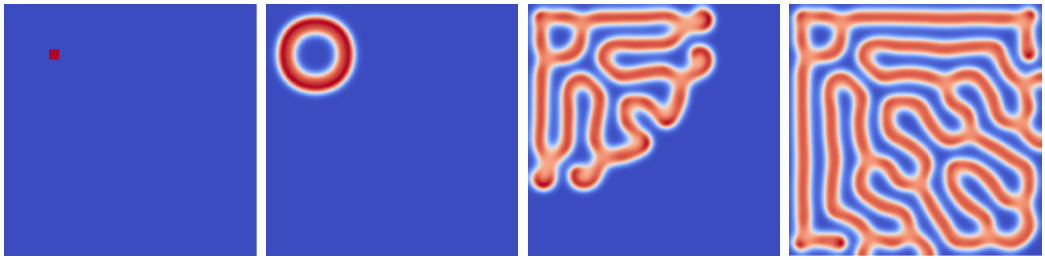
d_a: the rate of growth for a

d_b: the rate of growth for b

f: the feed rate also keeps a from exceeding 1

k: the kill rate of b which is scaled by b to prevent it from being less than 0

Figure 10: Coral Simulation



(a) $r = 0.1$ (b) $r = 0.2$ (c) $r = 0.3$ (d) $r = 0.3$

These patterns, also known as Turing patterns, can be used to generate anything from mitosis and spirals to zebrafish patterns. An example of the differences between Reaction and Reaction Diffusion

systems is, firstly, the spatial component, and, secondly, how the patterns change given their environment. In animal Turing patterns, the area and type of skin effects the pigment patterns. For leopards, the patterns of the patches are not homogeneous. Around the center of their sides, the patches are normally larger and decrease in size as they go to the paws or face (Liu et al., 2006).

6 Conclusion

Generality is both a blessing and a curse. Reaction Diffusion systems are incredibly broad. Trying to understand 'movement' and 'flow' in higher dimensions between multiple components is confusing, but once understood, their applications are endless. Calculus is at the core of Reaction Diffusion. Most of the equations are partial differential equations, and all of Reaction Diffusion contains the Laplace operator. Reaction equations are what govern the interactions between objects, which change over time using differential equations. Diffusion governs how reactions change over space in Reaction Diffusion systems. Emulsifications, disease, populations, patterns, and even things such

as the probably of particles in quantum systems can be modeled using reaction diffusion equations. The problem is, with such generality, understanding how and, more importantly, why these equations work is difficult.

References

Cahn, J. W. and Hilliard, J. E. (1958). Free energy of a nonuniform system. i. interfacial free energy. *The Journal of Chemical Physics*, 28.

F. J. Vermolen, e. a. (2009). Numerical solutions of some diffuse interface problems: The cahn-hilliard equation and the model of thomas and windle. *International Journal for Multiscale Computational Engineering*, 7:523–543.

Foryś and Marciniak-Czochra (2003). Logistic equation in tumour growth modelling. *Int. J. Appl. Math. Comput. Sci*, 13:317–325.

Gray and Scott (1984). Autocatalytic reactions in the isothermal, continuous stirred tank reactor: Oscillations and instabilities in the system a + 2b to 3b; b to c. *Chemical Engineering Science*, 39(6):1087–1097.

Jessica (2010). Reaction lamp #1.

Joon Choi, Y. and Anderson, P. (2012). Cahn–hilliard modeling of

particles suspended in two-phase flows. *International Journal for Numerical Methods in Fluids*, 69.

Liu, R. T., Liaw, S. S., and Maini, P. K. (2006). Two-stage turing model for generating pigment patterns on the leopard and the jaguar. *Phys. Rev. E*, 74:011914.

R.A.Fisher (1937). The wave of advance of advantageous genes. *Annals of Human Genetics*, 7:355–369.

Zhong, Wang, Zhengyi, and Lu (1999). Global stability for a lotka-volterra reaction-diffusion system with a qualitatively stable matrix. *Mathematical and Computer Modeling*, 30:55–62.

7 Appendix

List of Figures

1 Example functions 5

3 the gradient vector field: $\nabla * f(x, y)$ 8

4 Heat Diffusion Progression 11

5 Logistic equation $p_0 = 3$ $k = 100$ 12

6 Examples of the Lotka-Volterra Model 14

7 Examples of SIR model 16

8 Fisher Equation Example 18

9 Cahn-Hilliard Examples at $t = 2000$ 19

10 Coral Simulation 21

Listing 1: Surface

```
import numpy as np
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt

# base function
def fun(x, y):
    return x*(y**2) + y

```

```
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
x = y = np.arange(-2.0, 2.0, 0.01)
X, Y = np.meshgrid(x, y)
zs = np.array([dx(x, y) for x, y in zip(np.ravel(X), np.ravel(Y))])
Z = zs.reshape(X.shape)

ax.plot_surface(X, Y, Z)

ax.set_xlabel('X Label')
ax.set_ylabel('Y Label')
ax.set_zlabel('Z Label')

plt.show()
```

Listing 2: Gradient

```
import numpy as np
import matplotlib.pyplot as plt

# The base function
def fun(x, y):
    return x*(y**2) + y

def dy(x, y): return 2*x*y  # Partial derivative on Y
def dx(x, y): return 2*y+1  # Partial derivative on X

# gets input points
x = y = np.arange(-2.0, 2.0, 0.25)
```

```
X, Y = np.meshgrid(x, y)
# Derivatives for each point
Dx = np.array([dx(x, y) for x, y in zip(np.ravel(X), np.ravel(Y))])
Dy = np.array([dy(x, y) for x, y in zip(np.ravel(X), np.ravel(Y))])
Dx = Dx.reshape(X.shape)
Dy = Dy.reshape(Y.shape)
# Vectors given derivatives
Ax = np.array([x + dx for x, dx in zip(np.ravel(X), np.ravel(Dx))])
Ay = np.array([y + dy for y, dy in zip(np.ravel(Y), np.ravel(Dy))])
Ax = Ax.reshape(X.shape)
Ay = Ay.reshape(Y.shape)
# Graph
plt.quiver(X, Y, Ax, Ay, cmap=plt.cm.coolwarm)
plt.show()
```

Listing 3: Divergence

```
import matplotlib.pyplot as plt # graphing library
from numpy import fromfunction # generates matrix

size: int = 100 # size of the matrix

# The base function
def fun(x, y): return x*(y**2) + y

def dy(x, y): return 2*x*y # Partial derivative on Y
def dx(x, y): return 2*y+1 # Partial derivative on X

# Divergence function, sum of the partials
```

```
13 def div(x, y): return dx(x,y) + dy(x,y)
14
15 # transforms the coordinates to get the right domain
16 def t(x): return (x - (size/2))/10
17
18 # Generates a matrix from a functions that is given X and Y
19 # In this case, the x, y operate like a cartesian plane centered at
     0
20 # we then plug in our modified coordients into our div function
21 P = fromfunction(lambda x, y: div(t(x), t(y)), (size, size))
22
23 # Graphs the matrix
24 plt.imshow(P, cmap=plt.cm.coolwarm, interpolation='bilinear')
25 plt.colorbar() # side colorbar for reference
26 plt.show()
```

Listing 4: Diffusion

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 # Simulation constants
5 alpha = 0.00001  # Laplician multiplier
6 n = 8000  # number of iterations
7 s = 100  # size of the matrix
8 dx = 2/s  # Simulation speed/accuracy
9
10 # Matrix initialization
11 M = np.fill((s, s), 0.0)
```

```
# Hotspot Seeding
M[20:30, 45:50] = 90
M[70:80, 45:50] = 90

# Implementation of the decretized laplacian operator
def laplacian(Z):
    Ztop = Z[0:-2, 1:-1]
    Zleft = Z[1:-1, 0:-2]
    Zbottom = Z[2:, 1:-1]
    Zright = Z[1:-1, 2:]
    Zcenter = Z[1:-1, 1:-1]
    return (Ztop + Zleft + Zbottom + Zright - (4 * Zcenter)) / dx**2

# Iterations through PDE
for _ in range(n):
    Mn = M[1:-1, 1:-1]
    M[1:-1, 1:-1] = Mn + np.multiply(laplacian(M), alpha)

# Graph Data
plt.imshow(M, cmap=plt.cm.coolwarm,
           interpolation='bilinear', extent=[-1, 1, -1, 1])
plt.show()
```

Listing 5: Logistic Equation

```
import matplotlib.pyplot as plt  # Graphing lib

# Model Params
r = 0.3  # growth rate
```

```
k = 100  # carrying capacity
p = 3  # init value

# Simulation params
t = 50  # total time units
dt = 0.01  # the resiprical of the iterations per time
n = int(t/dt)  # number of iterations

# Storing calculated values
y = []  # Output
t = []  # time for graphing

# Solve ODE using Euler's method
for i in range(n):
    p += ((r * p) * (1 - (p/k))) * dt  # logistic equation
    y.append(p)  # records output
    t.append(i*dt)  # records time

# graphs points
plt.plot(t, y)
plt.show()
```

Listing 6: Lotka-Volterra

```
import matplotlib.pyplot as plt

# Model params
alpha = 2/3
beta = 4/3
```

```
gamma = 1
delta = 1

# Simulation constants: descrete time and space
dt = 0.001  # the change in time
n = 20000  # number of iterations

# Seed the space
U = 1  # Population of Pray
V = 0.1  # Population of Preditors
# output Arrays
Uy = []
Vy = []
X = []

# Simulate this ODE with Euler's method
for i in range(n):
    # Gets copies to prevent overwritting
    Uc = U
    Vc = V
    # Generate next time step
    U += (alpha*Uc - beta*Uc*Vc)*dt
    V += (gamma*Uc*Vc - delta*Vc)*dt
    # Record outputs
    Uy.append(U)
    Vy.append(V)
    X.append(i*dt)

```

```
# Graph Data
plt.title("Lotka-Volterra Model")
plt.plot(X, Uy, 'b', label="pray")
plt.plot(X, Vy, 'r', label="preditor")
plt.legend()
plt.show()
```

Listing 7: SIR

```
import matplotlib.pyplot as plt

# Model params
b = 1  # days of contact to spread
k = 1/6  # days to recover

# Simulation params
dt = 0.1  # change in time
n = 800  # number of iterations

# Seeding
s = 1  # suseptible population
i = 1.27E-4  # Infected
r = 0  # Recovered
# To graph and record values
Sy = []
Iy = []
Ry = []
X = []

```

```
# Simulate ODE with Euler's method
for x in range(n):
    # Edge case
    if i <= 0:
        print("Done")
        break
    # copy to prevent overwritting
    Sc = s
    Ic = i
    Rc = r
    # Generate next timestep
    s += (-b * Sc * Ic) * dt
    i += ((b * Sc * Ic) - (k * Ic)) * dt
    r += (k * Ic) * dt
    # Record values
    Sy.append(s)
    Iy.append(i)
    Ry.append(r)
    X.append(x*dt)

# Graph Data
plt.title("SIR model")
plt.plot(X, Sy, 'g', label="Suseptible")
plt.plot(X, Iy, 'r', label="Infected")
plt.plot(X, Ry, 'b', label="Recovered")
plt.legend()
plt.show()
```

Listing 8: Cahn-Hilliard

```
from scipy.ndimage.filters import laplace  # Descrete Laplace
import matplotlib.pyplot as plt  # Graphing
import numpy.random as r  # Matrix Seeding

# Simulation constants
alpha = 0.04  # Diffusion constant
gamma = 0.125  # constant multiplier
n = 2000  # number of iterations
s = 200  # size of the matrix

# Matrix initialization
r.seed(100)
M = r.rand(s, s)

# iterating the Cahn-Hilliard Equation
for _ in range(n):
    G = gamma * laplace(M)  # inner laplace
    Mu = M**3 - M - G  # Mu: chemical potential
    M += laplace(Mu) * alpha  # outer laplace

# Graph Data
plt.imshow(M, cmap=plt.cm.coolwarm, interpolation='bicubic')
plt.axis('off')
plt.show()
```

Listing 9: Gray-Scott

```
from scipy.ndimage.filters import laplace  # Descrete Laplace
import matplotlib.pyplot as plt  # graphing
from numpy import full  # matrix init

# Constants
s = 250  # size of the matrix
Du = 1  # U growth rate
Dv = 0.5  # V growth rate
f = .0545  # feed rate
k = .062  # kill rate


# Simulation Params
t = 5000  # 'time' simulated
dt = .1  # Iterations per time unit
n = int(t/dt)  # total iteration count

fig = plt.figure()

# Matrix Inits
U = full((s, s), 1.0)  # Starts U full of 1
V = full((s, s), 0.0)  # stats V full of 0
V[45:55, 45:55] = 1.0  # Adds a bit to V

# iterate though PDE
for _ in range(n):
    uvv = U * V * V  # prevents write error and clean
    U += ((Du*laplace(U) - uvv + (f * (1 - U))) * dt)  # U solve
```

```
29      V += ((Dv*laplace(V) + uvv - (V * (f + k))) * dt)  # V solve
30
31  # Graph
32  plt.imshow(V, cmap=plt.cm.coolwarm)
33  plt.axis("off")
34  plt.show()
```