Reaction diffusion systems are mathematical models which correspond to many physical phenomena. The systems describe phenomena in chemistry, geology, and physics. We will study a nondimensional and scaled reaction diffusion system of equations. The general form of the system is

$$\frac{\partial u}{\partial t} = \gamma f(u,v) + \nabla^2 u \tag{1}$$

$$\frac{\partial y}{\partial t} = \gamma g(u,v) + \mathrm{d}\nabla^2 v \tag{2}$$

where u is the activator, v is the inhibitor, d is the ratio of diffusion coefficients, and γ is a special scaping factor [1]. The functions $f(u,v)$ and $g(u,v)$ describe the reaction kinetics for the activator and inhibitor. Here are three other nondimensionalized reaction systems:

$$f(u,v) = a - u - h(u,v),\ \ g(u,v) = \alpha(b - v) - h(u,v) \tag{3}$$

$$h(u,v) = \frac{puv}{1 + u + Ku^2}$$

We will numerically solve system (3) using python. Our initial conditions will consist of random perturbations from the steady state values of $u_s = 9$ and $v_s = 10$ by adding a random value to each number respectively as shown by the code. (Appendix A) Our parameters are also randomly generated. Our parameters for figure the results in figure 3 and figure 4 are in Appendix C and D respectively.

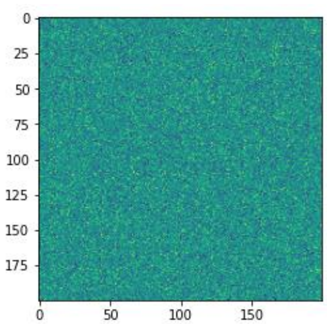

Figure 1: Initial state of the reaction diffusion system with a diffusivity constant of 15

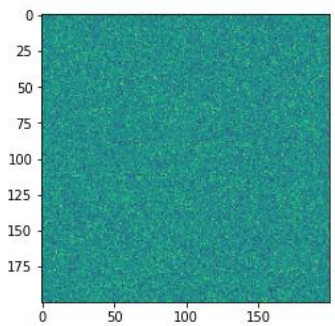

Figure 2: Initial state of the reaction diffusion system with a diffusivity constant of 6

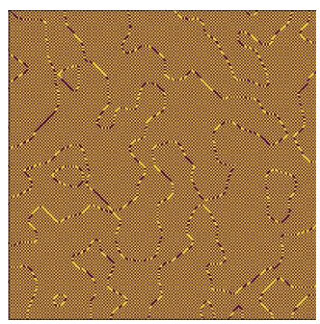

Figure 3: Solution of the reaction diffusion system with a diffusivity constant of 15

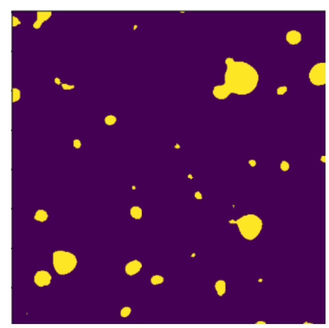

Figure 4: Solution of the reaction diffusion system with a diffusivity constant of 6

There were two different types of graphical results for the system of equations. As seen in figures 1 and 2, both results began at extremely similar initial states. However, as seen in figures 3 and 4, their final states differed completely, while one resembles a maze-like pattern the other resembles a polka-dot pattern. Since we randomly generated both our initial conditions and our parameters, they are most likely the culprits of this discrepancy. After noticing this difference, care was taken to record the parameters during each run. After observing the percent difference in the parameters through many trials (figure 5), it was discovered that when two solutions differed as seen in the figures, the percent difference in diffusivity was 67% (Appendix B) While this factor was isolated, there may be other factors at play. This would require more research and a better understanding of reaction diffusion systems. In further experiments, I would examine the initial conditions to find patterns where different solution patterns are formed.

Appendix A:

Python Code

In [1]:

```
import time
import matplotlib.pyplot as plt
import matplotlib.animation as animation
from scipy import signal
import numpy as np
from numpy import random
import random

# Parameters
#Da = 1
#-------
#d = 10
#f = 0.05
#k = 0.5
#dt = 1
#p = 18.5
#alpha = 0.5
#b = 9
#K = 0.1
#a = 3
#-------------------------------------------------------------------------------
# -------------------------Super Random Paramenters:--------------------------
# Create Empty Parameter Array
par = np.zeros((1,9))
print(par)
# Create Parameters Using Random Numbers and the Index
for i in range(1,10):
    if i % 2 == 0:
        par[:,i-1] =  abs(i + (1/(i+1)) * np.random.normal(0.5,1))/50
    else:
        par[:,i-1] =  abs(i - (1/(i+1)) * np.random.normal(0.5,1))*10

# Print Parameter Array for Possible Analysis
print(par)
# Define Parameters
d = par[0,0]
#f = par[0,1]
k = par[0,2]
dt = 0.1 #par[0,3] Keep dt Constant
p = par[0,4]
alpha = par[0,5]
b = par[0,6]
K = par[0,7]
a = par[0,8]
#-------------------------------------------------------------------------------

# Laplacian kernel
L = np.array([[0.05, 0.2, 0.05], [0.2, -1, 0.2], [0.05, 0.2, 0.05]])

class Skin:
    # Input variables for the skin
    m = 250       # skin will be m by m where m = skin size

    # Initialize the skin
    #a = np.ones([m, m])   # activator
    #b = np.zeros([m, m])  # inhibitor
    #------------------------- Randomized Skin ------------------------------------------------------
    a = 10 + 0.5 * np.random.randn(m,m) # Activator
    b = 9 + 0.5 * np.random.randn(m,m) # Inhibitor
    b[int(m / 20 - 1):int(m / 20 + 1), int(m / 20 - 1):int(m / 20 + 1)] = 1
    pat = a - b  #pattern

 # h(u,v) function
def h(u,v):
    return (p*u*v)/(1+u+K*(u**2))
```

```
# Update the pattern based on the reaction-diffusion system, each call to update_skin is one time step
def update_skin(sk): # sk.a = u, sk.b = v
    La = signal.convolve(sk.a, L, mode='same')
    Lb = signal.convolve(sk.b, L, mode='same')
    #an = sk.a + dt * (Da * La - sk.a * sk.b**2 + f * (1 - sk.a))
    #bn = sk.b + dt * (Db * Lb + sk.a * sk.b**2 - (k + f) * sk.b)
    # Represents f(u,v):
    an = sk.a + dt * (La - (a - sk.a - h(sk.a,sk.b)))
    # Represents g(u,v):
    bn = sk.b + dt * (d * Lb + alpha * (b-sk.b) - h(sk.a,sk.b))
    sk.a = an
    sk.b = bn
    im = an - bn
    return im

my_skin = Skin()

##### Animate the pattern #####

# Required line for plotting the animation
%matplotlib notebook
# Initialize the plot of the skin that will be used for animation
fig = plt.gcf()
# Show first image - which is the initial pattern
im = plt.imshow(my_skin.pat)
plt.show()

# Helper function that updates the pattern and returns a new image of
# the updated pattern. animate is the function that FuncAnimation calls
def animate(frame):
    im.set_data(update_skin(my_skin))
    return im,

# This line creates the animation
anim = animation.FuncAnimation(fig, animate, frames=5000,
                               interval=1)
```

Appendix B:

Percent Difference Between Solutions

```
[[69.32922439]   [['d']
 [ 0.50681233]    ['f']
 [15.94137006]    ['k']
 [ 0.        ]    ['dt']
 [ 6.73500442]    ['p']
 [ 7.6278565 ]    ['alpha']
 [ 0.2654519 ]    ['b']
 [ 3.2518336 ]    ['K']
 [ 2.77232363]]   ['a']]
```

Percent difference between parameters in solutions. Diffusivity has a 67% difference.

Appendix C

Parameters for Solutions

```
[[4.15820672e+00]   [['d']
 [4.18338654e-02]    ['f']
 [2.37065956e+01]    ['k']
 [1.00000000e-01]    ['dt']
 [4.58589069e+01]    ['p']
 [1.25432397e-01]    ['alpha']
 [6.76849082e+01]    ['b']
 [1.62935476e-01]    ['K']
 [8.96428263e+01]]   ['a']]
```

Parameters for solution in figure 3

```
[[1.35575532e+01]   [['d']
 [4.20469646e-02]    ['f']
 [2.82024530e+01]    ['k']
 [1.00000000e-01]    ['dt']
 [4.91705453e+01]    ['p']
 [1.16542688e-01]    ['alpha']
 [6.75057130e+01]    ['b']
 [1.57803954e-01]    ['K']
 [8.72246760e+01]]   ['a']]
```

Parameters for solution in figure 3

References:

https://en.wikipedia.org/wiki/Reaction%E2%80%93diffusion_system