Numerical Solutions to Reaction Diffusion Equations

Christian Frederiksen and Eric Sung

1 Introduction

The diffusion equation appears in many mathematical models. It is used not only in models of physical phenomenon, like heat transfer or the diffusion of a solute, but also in economic models and models of population dynamics. Adding a reaction terms results in the reaction-diffusion equation which in its most general form is given by:

$$\frac{\partial u}{\partial t} = \mu \Delta u + f(t, x, u) \tag{1}$$

Here Δu is the Laplacian of u, μ is a constant diffusivity, and $f(t, x, u)$ is a reaction term. When $f(t, x, u) = 0$, (1) becomes the diffusion equation. The reaction term can be used to model many different phenomenon like heat sources, the addition of a solute, or chemical reactions. Some specific reaction diffusion equations are:

1. Fisher's Equation

$$\frac{\partial u}{\partial t} = \mu \Delta u + u(1 - u)$$

2. Newell-Whitehead-Segel Equation

$$\frac{\partial u}{\partial t} = \mu \Delta u + u(1 - u^2)$$

3. Zeldovich Equation

$$\frac{\partial u}{\partial t} = \mu \Delta u + u(1 - u)(u - a)$$

Due to the prevalence of diffusion and reaction diffusion equations solving them is important to a number of sciences. One means of solving reaction-diffusion equations is with finite difference methods. In this paper we will outline finite difference methods which can be used to solve reaction-diffusion equations for a number of different boundary conditions in both one and two dimensions. All of the described methods have been implemented in Matlab and we will include numerous examples displaying how solutions to (1) evolve over time and the error of the methods used.

2 Diffusion Equation in One Dimension

In one dimension the diffusion equation is given by:

$$\frac{\partial u}{\partial t} = \mu \frac{\partial^2 u}{\partial x^2} \tag{2}$$

In order to solve this numerically we used the Crank-Nicolson method, a second order implicit finite difference method. In one dimension the method applied to (2) is given by:

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \frac{\mu}{2} \frac{u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}}{(\Delta x)^2} + \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{(\Delta x)^2}$$

Setting $r = \frac{\mu \Delta t}{2(\Delta x)^2}$ we can rearrange the above to obtain

$$(1 + 2r)u_i^{n+1} - r(u_{i-1}^{n+1} + u_{i+1}^{n+1}) = (1 - 2r)u_i^n + r(u_{i-1}^n + u_{i+1}^n) \tag{3}$$

This leaves a linear system which may be solved directly for u^{n+1} at each timestep. The exact form of the system depends on the given boundary conditions. In the next sections we give the form of the system and its solution for Dirichlet, Neumann, and periodic boundary conditions.

2.1 Dirichlet Boundary Conditions

Suppose we are given an interval $[0, L]$ and we wish to find a solution to (2) given the boundary conditions $u(0) = b_0$ and $u(L) = b_1$. First we divide $[0, L]$ into $N + 1$ equal subintervals. This leaves $N + 2$ nodes $x_0, x_1, ..., x_N, x_{N+1}$ however because we know $u_0 = u(0) = b_0$ and $u_{N+1} = u(L) = b_1$ for all t there are N unknowns at each timestep: $x_1, ..., x_N$. Considering (3) at x_1 and x_N we have

$$
\begin{aligned}
&(1+2r)u_1^{n+1} - r(u_0^{n+1} + u_2^{n+1}) = (1-2r)u_1^n + r(u_0^n + u_2^n) \\
\Rightarrow \quad &(1+2r)u_1^{n+1} - ru_2^{n+1} = (1-2r)u_1^n + ru_2^n + 2rb_0 \\
&(1+2r)u_N^{n+1} - r(u_{N-1}^{n+1} + u_{N+1}^{n+1}) = (1-2r)u_N^n + r(u_{N-1}^n + u_{N+1}^n) \\
\Rightarrow \quad &(1+2r)u_N^{n+1} - ru_{N-1}^{n+1} = (1-2r)u_N^n + ru_{N-1}^n + 2rb_1
\end{aligned}
$$

and considering (3) at x_j for $j = 2, 3, ..., N-1$, it remains unchanged. This gives us the linear system:

$$
\begin{bmatrix}
1+2r & -r & 0 & \dots & 0 & 0 & 0 \\
-r & 1+2r & -r & \dots & 0 & 0 & 0 \\
0 & -r & 1+2r & \dots & 0 & 0 & 0 \\
\vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\
0 & 0 & 0 & \dots & 1+2r & -r & 0 \\
0 & 0 & 0 & \dots & -r & 1+2r & -r \\
0 & 0 & 0 & \dots & 0 & -r & 1+2r
\end{bmatrix}
\begin{bmatrix}
u_1^{n+1} \\ u_2^{n+1} \\ u_3^{n+1} \\ \vdots \\ u_{N-2}^{n+1} \\ u_{N-1}^{n+1} \\ u_N^{n+1}
\end{bmatrix}
=
\begin{bmatrix}
(1-2r)u_1^n + ru_2^n + 2rb_0 \\
(1-2r)u_2^n + r(u_1^n + u_3^n) \\
(1-2r)u_3^n + r(u_2^n + u_4^n) \\
\vdots \\
(1-2r)u_{N-2}^n + r(u_{N-3}^n + u_{N-1}^n) \\
(1-2r)u_{N-1}^n + r(u_{N-2}^n + u_N^n) \\
(1-2r)u_N^n + ru_{N-1}^n + 2rb_1
\end{bmatrix}
$$

This system is tridiagonal and is efficiently solved using the Thomas Algorithm which is what we implemented in our code.

2.1.1 Example 1: 1D Diffusion with Dirichlet Boundary Conditions

Below is a plot of the output of our code for the 1D diffusion equation with Dirichlet boundary conditions for 1, 20, 40, and 60 timesteps with a timestep size of 0.001. The initial condition was set to $g(x) = \sin(6\pi x) + x - 1$ and the domain, [0,1], was divided into 128 equally spaced subintervals for the method.

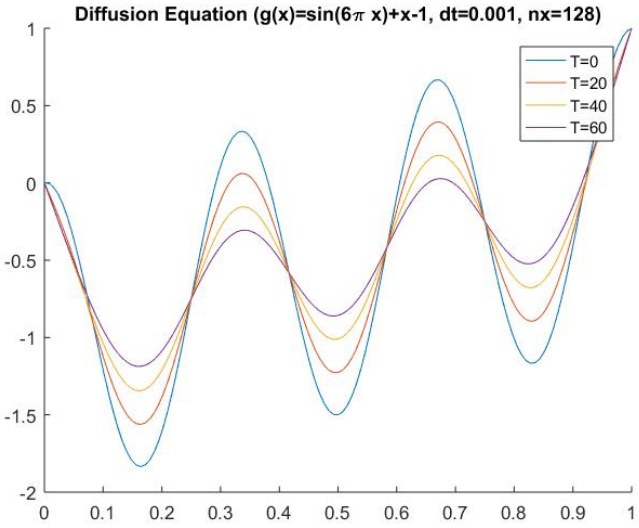

Figure 1: Solution of Example 2 over Time

In order to check the accuracy of the method we wrote code which repeated the above calculations at six different discretization levels. We set the subinterval and timestep sizes to be equal to $h = 1/16, 1/32, 1/64, 1/128, 1/256, 1/512$. Then we ran our code for each value of h until time .0625 and stored the resulting solution. We then approximated the error of the method by setting $e_n = |x_n - x_{512}|$ where x_n is the computed solution at x when using $h = 1/n$. To check the order of accuracy we computed $f_n = e_n/e_{n+1}$ which yielded the following data:

```
f =

  Columns 1 through 10

       NaN    3.2311    3.0548    7.8214    3.1436    6.3315    6.4898    6.4560    6.4742    6.4560
       NaN    6.3733    2.1023    2.3924    3.4870    4.6795    4.7262    4.6924    4.7025    4.6924
       NaN    3.7967    4.6755    4.2858    3.9633    4.3645    4.3831    4.3692    4.3723    4.3692
       NaN    5.0940    5.0421    4.9845    4.9401    5.0469    5.0555    5.0505    5.0515    5.0505
       NaN       Inf       Inf       Inf       Inf       Inf       Inf       Inf       Inf       Inf

  Columns 11 through 17

    6.4898    6.3315    3.1436    7.8214    3.0548    3.2311       NaN
    4.7262    4.6795    3.4870    2.3924    2.1023    6.3733       NaN
    4.3831    4.3645    3.9633    4.2858    4.6755    3.7967       NaN
    5.0555    5.0469    4.9401    4.9845    5.0421    5.0940       NaN
       Inf       Inf       Inf       Inf       Inf       Inf       NaN
```

Figure 2: Relative Error of Example 2

We see that $f_n \approx 4$ for most x and n which is what we would expect for a second order method since the difference in h was by a factor of two.

Below are two visualizations of the long term behavior of example 2. The first figure shows the solution at specified values of time where we see the solution trending towards a linear plot between the two boundary conditions. The second figure shows the evolution continuously over time. The left plot shows only the first 200 timesteps to increase the resolution around the initial condition while the right shows the first 5000 timesteps.

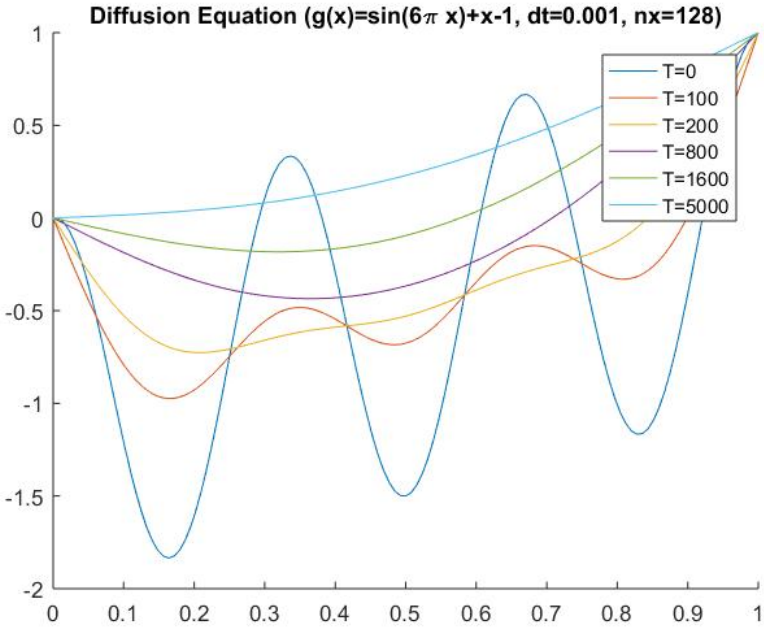

Figure 3: Long term solution behavior of example 2

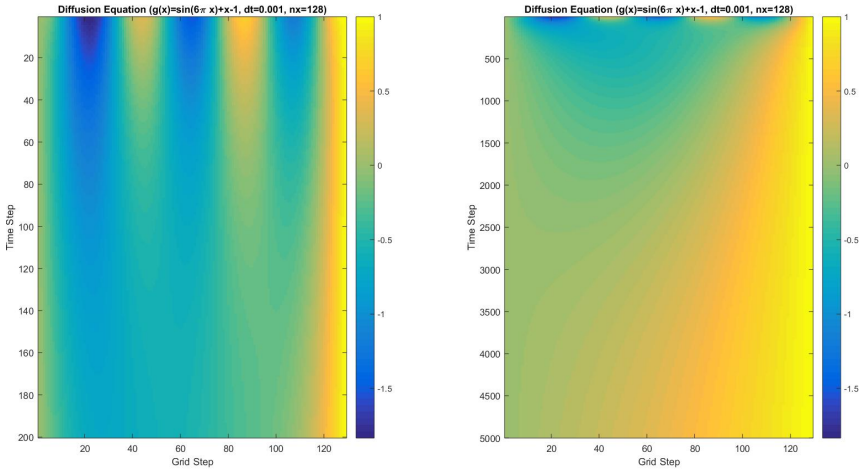

Figure 4: Long term solution behavior of example 2

2.2 Neumann Boundary Conditions

Suppose we are given an interval $[0, L]$ and we wish to find a solution to (2) given the boundary conditions $u'(0) = c_0$ and $u'(L) = c_1$. First we divide $[0, L]$ into N equal subintervals. This leaves $N + 1$ nodes $x_0, x_1, ..., x_N$ which we must solve for u at at every timestep. In order to satisfy the Neumann boundary condition we create two "ghost nodes" at $x_{-1} = x_0 - \Delta x$ and $x_{N+1} = x_N + \Delta x$ whose values we evaluate using the boundary conditions:

$$c_0 = \frac{\partial u}{\partial x}|_{x=0} \approx \frac{u_1 - u_{-1}}{2\Delta x} \qquad \Rightarrow \qquad u_{-1} \approx u_1 - 2c_0\Delta x$$
$$c_1 = \frac{\partial u}{\partial x}|_{x=L} \approx \frac{u_{N+1} - u_N}{2\Delta x} \qquad \Rightarrow \qquad u_{N+1} \approx u_N + 2c_1\Delta x$$

Then considering (3) at x_0 and x_N we have:

$$(1 + 2r)u_0^{n+1} - r(u_{-1}^{n+1} + u_1^{n+1}) = (1 - 2r)u_0^n + r(u_{-1}^n + u_1^n)$$
$$\Rightarrow \qquad (1 + 2r)u_0^{n+1} - 2ru_1^{n+1} = (1 - 2r)u_0^n + 2ru_1^n - 4rc_0\Delta x$$
$$(1 + 2r)u_N^{n+1} - r(u_{N-1}^{n+1} + u_{N+1}^{n+1}) = (1 - 2r)u_N^n + r(u_{N-1}^n + u_{N+1}^n)$$
$$\Rightarrow \qquad (1 + 2r)u_N^{n+1} - 2ru_{N-1}^{n+1} = (1 - 2r)u_N^n + 2ru_{N_1}^n + 4rc_1\Delta x$$

and considering (3) at x_j for $j = 2, 3, ..., N - 1$, it remains unchanged. This gives rise to the linear system:

$$\begin{bmatrix} 1+2r & -2r & 0 & \dots & 0 & 0 & 0 \\ -r & 1+2r & -r & \dots & 0 & 0 & 0 \\ 0 & -r & 1+2r & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1+2r & -r & 0 \\ 0 & 0 & 0 & \dots & -r & 1+2r & -r \\ 0 & 0 & 0 & \dots & 0 & -2r & 1+2r \end{bmatrix} \begin{bmatrix} u_0^{n+1} \\ u_1^{n+1} \\ u_2^{n+1} \\ \vdots \\ u_{N-2}^{n+1} \\ u_{N-1}^{n+1} \\ u_N^{n+1} \end{bmatrix} = \begin{bmatrix} (1-2r)u_0^n + 2ru_1^n - 4r\Delta x\sigma_a \\ (1-2r)u_1^n + r(u_0^n + u_2^n) \\ (1-2r)u_2^n + r(u_1^n + u_3^n) \\ \vdots \\ (1-2r)u_{N-2}^n + r(u_{N-3}^n + u_{N-1}^n) \\ (1-2r)u_{N-1}^n + r(u_{N-2}^n + u_N^n) \\ (1-2r)u_N^n + 2ru_{N-1}^n + 4r\Delta x\sigma_b \end{bmatrix}$$

The system above is tridiagonal and is efficiently solved using the Thomas Algorithm which is what we implemented in our code.

2.2.1 Example 2: 1D Diffusion with Neumann Boundary Conditions

Below is a plot of the output of our code for the 1D diffusion equation with Neumann boundary conditions for 1, 20, 40, and 60 timesteps with a timestep size of 0.001. The initial condition was set to $g(x) = 3x + 3\sin(6\pi x)$ and the domain, [0,1], was divided into 128 equally spaced subintervals for the method.

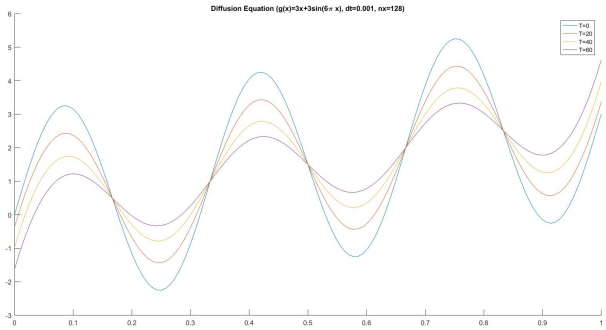

Figure 5: Solution of Example 3 over time

We calculated the accuracy of the method as in example 1 from Section 2.1.1. As in the first example we see that $_n \approx 4$ for most x and n which confirms we were getting approximately second order accuracy with the method. The data for this computation was taken at time 0.0625 which is approximately the final solution shown in figure 7.

```
f =

  Columns 1 through 10

    6.7424    1.2777    2.4030    3.9975    5.9497    6.4332    6.4481    6.4706    1.0000    6.4706
    3.3840    5.5062    4.4258    3.8453    4.5119    4.7673    4.6838    4.7009    0.9984    4.7009
    3.8025    3.8625    4.2139    4.1807    4.2967    4.4037    4.3653    4.3719    1.0113    4.3719
    4.7677    5.0262    5.0044    4.9981    5.0267    5.0612    5.0490    5.0514    1.0401    5.0514
       Inf       Inf       Inf       Inf       Inf       Inf       Inf       Inf       Inf       Inf

  Columns 11 through 17

    6.4481    6.4332    5.9497    3.9975    2.4030    1.2777    6.7424
    4.6838    4.7673    4.5119    3.8453    4.4258    5.5062    3.3840
    4.3653    4.4037    4.2967    4.1807    4.2139    3.8625    3.8025
    5.0490    5.0612    5.0267    4.9981    5.0044    5.0262    4.7677
       Inf       Inf       Inf       Inf       Inf       Inf       Inf
```

Figure 6: Relative Error for Example 3

Below are two visualizations of the long term behavior of example 3. The first figure shows the solution at specified values of time where we see the solution trending towards a linear plot between the two boundary conditions. The second figure shows the evolution continuously over time. The left plot shows only the first 200 timesteps to increase the resolution around the initial condition while the right shows the first 5000 timesteps.

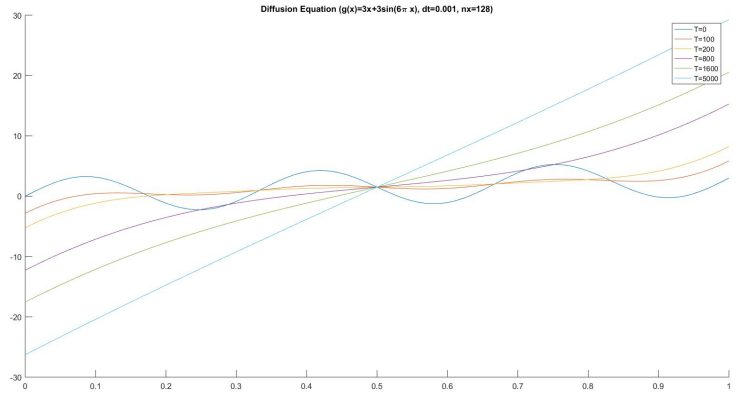

Figure 7: Long term solution behavior of example 3

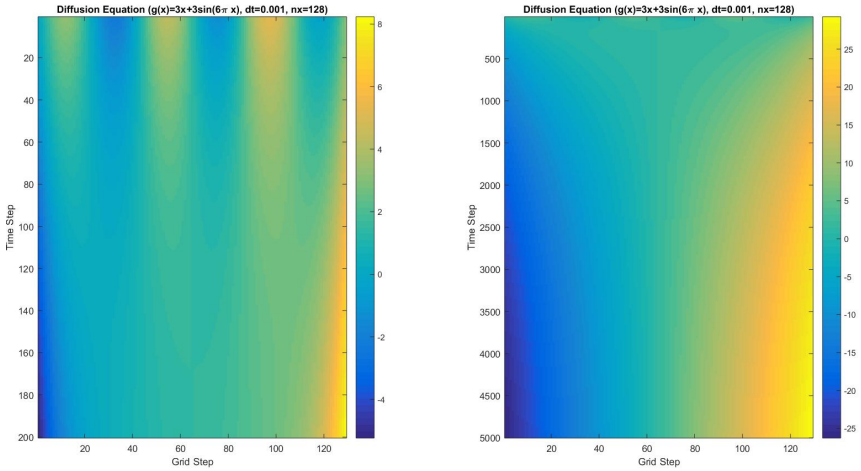

Figure 8: Long term solution behavior of example 3

2.3 Periodic Boundary Conditions

Suppose we are given an interval $[0, L]$ and we wish to find a solution to (2) given periodic boundary conditions: $u(0) = u(L)$ and $u'(0) = u'(L)$. First we divide $[0, L]$ into N equal subintervals. This leaves $N + 1$ nodes $x_0, x_1, ..., x_N$ but because $u(0) = u(L)$ there are only N unknowns: $x_1, x_2, ..., x_N$ to solve for at each timestep. Considering (3) at x_1 and x_N we have:

$$
\begin{aligned}
&(1+2r)u_1^{n+1} - r(u_0^{n+1} + u_2^{n+1}) = (1+2r)u_1^n - r(u_0^n + u_2^n) \\
\Rightarrow \quad &(1+2r)u_1^{n+1} - r(u_N^{n+1} + u_2^{n+1}) = (1+2r)u_1^n - r(u_N^n + u_2^n) \\
&(1+2r)u_N^{n+1} - r(u_{N-1}^{n+1} + u_{N+1}^{n+1}) = (1+2r)u_N^n - r(u_{N-1}^n + u_{N+1}^n) \\
\Rightarrow \quad &(1+2r)u_N^{n+1} - r(u_{N-1}^{n+1} + u_1^{n+1}) = (1+2r)u_N^n - r(u_{N-1}^n + u_1^n)
\end{aligned}
$$

and considering (3) at x_j for $j = 2, 3, ..., N-1$, it remains unchanged. This gives rise to the linear system:

$$
\begin{bmatrix}
1+2r & -r & 0 & \dots & 0 & 0 & -r \\
-r & 1+2r & -r & \dots & 0 & 0 & 0 \\
0 & -r & 1+2r & \dots & 0 & 0 & 0 \\
\vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\
0 & 0 & 0 & \dots & -r & 1+2r & -r \\
-r & 0 & 0 & \dots & 0 & -r & 1+2r
\end{bmatrix}
\begin{bmatrix}
u_1^{n+1} \\ u_2^{n+1} \\ u_3^{n+1} \\ \vdots \\ u_{N-1}^{n+1} \\ u_N^{n+1}
\end{bmatrix}
=
\begin{bmatrix}
(1-2r)u_1^n + r(u_N^n + u_2^n) \\
(1-2r)u_2^n + r(u_1^n + u_3^n) \\
(1-2r)u_3^n + r(u_2^n + u_4^n) \\
\vdots \\
(1-2r)u_{N-1}^n + r(u_{N-2}^n + u_N^n) \\
(1-2r)u_N^n + r(u_{N-1}^n + u_1^n)
\end{bmatrix}
$$

Since the system is no longer tridiagonal the Thomas algorithm will no longer be effective. Instead we found the LU factorization of the coefficient matrix and then solved the upper and lower tridiagonal systems.

2.3.1 Example 3: 1D Diffusion with Periodic Boundary Conditions

Below is a plot of the output of our code for the 1D diffusion equation with periodic boundary conditions for 1, 20, 40, and 60 timesteps with a timestep size of 0.001. The initial condition was set to $g(x) = 2\cos(4\pi x) + 4\sin(6\pi x)$ and the domain, [0,1], was divided into 128 equally spaced subintervals.

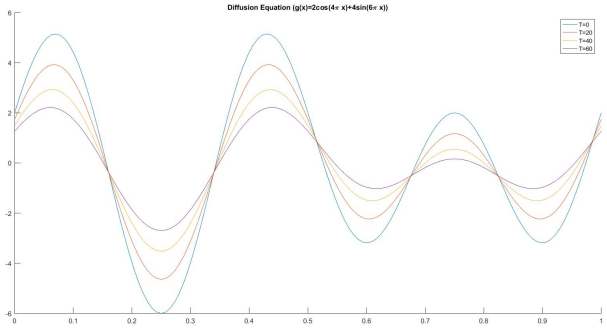

Figure 9: Solution of Example 4 over time

We calculated the accuracy of the method as in example 1 of Section 2.1.1. As in the first example we see that $_n \approx 4$ for most x and n which confirms we were getting approximately second order accuracy with the method. The data for this computation was taken at time 0.0625 which is approximately the final solution shown in figure 7.

```
f =

  Columns 1 through 10

    4.5638    4.5643    7.8011    4.5641    4.5642    4.5641    7.8010    4.5643    4.5638    4.5634
    4.1765    4.1766    4.8286    4.1765    4.1766    4.1765    4.8284    4.1766    4.1765    4.1764
    4.2328    4.2328    4.3985    4.2328    4.2328    4.2328    4.3977    4.2328    4.2328    4.2328
    5.0097    5.0097    5.0688    5.0097    5.0097    5.0097    5.0652    5.0097    5.0097    5.0097
       Inf       Inf       Inf       Inf       Inf       Inf       Inf       Inf       Inf       Inf

  Columns 11 through 17

    7.8008    4.5636    4.5635    4.5636    7.8007    4.5634    4.5638
    4.8277    4.1765    4.1764    4.1765    4.8275    4.1764    4.1765
    4.3950    4.2328    4.2328    4.2328    4.3943    4.2328    4.2328
    5.0522    5.0097    5.0097    5.0097    5.0487    5.0097    5.0097
       Inf       Inf       Inf       Inf       Inf       Inf       Inf
```

Figure 10: Relative Error for Example 4

Below are two visualizations of the long term behavior of example 4. The first figure shows the solution at specified values of time where we see the solution trending towards zero. The second figure shows the evolution continuously over time. The left plot shows only the first 200 timesteps to increase the resolution around the initial condition while the right shows the first 5000 timesteps.

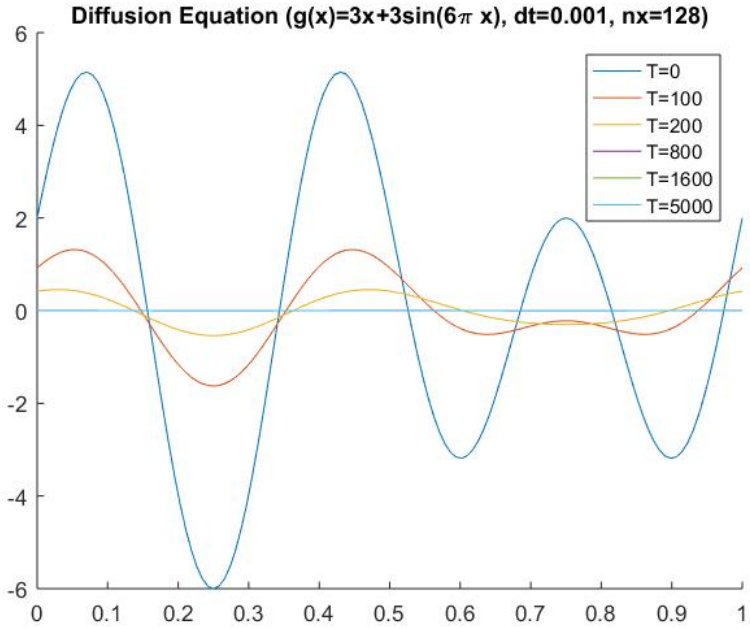

Figure 11: Long term solution behavior of example 4

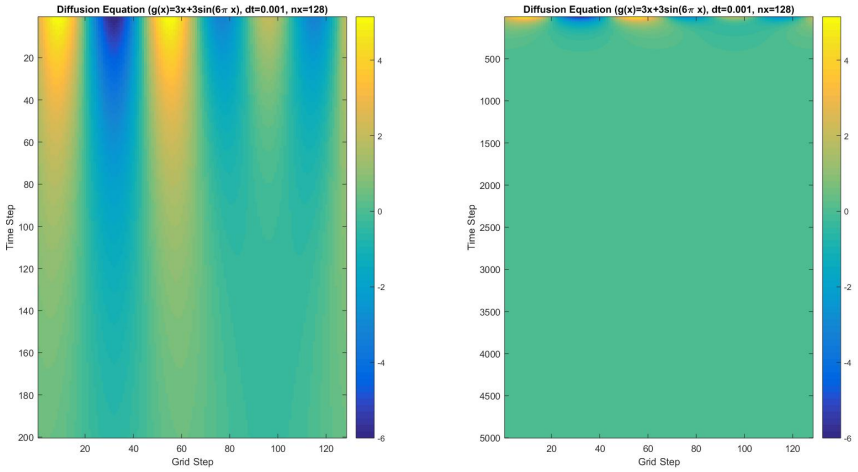

Figure 12: Long term solution behavior of example 4

3 Reaction-Diffusion Equation in One Dimension

In one dimension the reaction diffusion equation in its most general form is given by:

$$\frac{\partial u}{\partial t} = \mu \frac{\partial^2 u}{\partial x^2} + f(t, x, u) \tag{4}$$

The reaction term can result in a nonlinear system depending on how it is introduced in the numerical method. For this reason finding solutions to the reaction-diffusion equation can be much more complicated than finding solutions to the diffusion equation. We implemented both implicit and explicit methods of adding in the reaction term but ultimately our final codes all used an explicit method for adding the reaction term. We outline both the implicit and explicit methods which we implemented below.

3.1 Implicit Methods

One option for solving the one-dimensional reaction diffusion equation is to extend the Crank-Nicolson method. This method is implicit and takes the form:

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \frac{\mu}{2} \frac{u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}}{(\Delta x)^2} + \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{(\Delta x)^2} + \frac{f_i^{n+1}}{2} + \frac{f_i^n}{2}$$

where $f_i^n = f(t_{n+1}, x_i, u_i)$. Observe that the introduction of the term f_i^{n+1} results in the system becoming nonlinear so at each timestep we must use an additional method to solve the nonlinear system. Newton's method is one option for this and takes the form:

$$u_{j+1}^n = u_j^n + du \qquad \text{where} \qquad [J(u^n)] * du = -h(u^n)$$

$$[J(u^n)] = \begin{bmatrix} \frac{\partial h_1}{u_1}(u^n) & \frac{\partial h_1}{\partial u_2}(u^n) & \dots \\ \frac{\partial h_2}{\partial u_1}(u^n) & \frac{\partial h_2}{\partial u_2}(u^n) & \dots \\ \vdots & \vdots & \end{bmatrix} \qquad \text{and} \qquad h(u) = \begin{bmatrix} h_1(u^n) \\ h_2(u^n) \\ \vdots \end{bmatrix}$$

and h_j is given by:

$$h_j(u^{n+1}) = \frac{u_i^{n+1} - u_i^n}{\Delta t} - \frac{\mu}{2} \frac{u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}}{(\Delta x)^2} - \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{(\Delta x)^2} - \frac{f_i^{n+1}}{2} - \frac{f_i^n}{2}$$

Since the solution should evolve smoothly, using the current value of u as an initial guess u^0 is fairly efficient when timesteps are small. In our code we found that Newton's method tended to converge in a small (less than 10) number of iterations in most cases when using the current value of u as the initial guess.

3.2 Explicit Methods

An alternative to using Newton's method or another means of solving nonlinear systems is to use an explicit method for the reaction term. One such method which we implemented was the two-step Adams-Bashforth method which uses the current and previous timesteps to estimate the forward timestep. In our implementation of the code in Matlab we initialized the first two timesteps to the same value so that we would have a previous value to use. In one dimension the reaction diffusion equation using Adams-Bashforth for the reaction term is given by:

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \frac{\mu}{2}\frac{u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}}{(\Delta x)^2} + \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{(\Delta x)^2} + \frac{3}{2}f_i^n - \frac{1}{2}f_i^{n-1} \tag{5}$$

Since the reaction term in the above method can be calculated explicitly for each node and timestep, the reaction term can simply be added to the rest of the known information before solving the systems given in sections 2.1, 2.2, and 2.3. As a result finding the solution to the reaction-diffusion equation is nearly identical to finding the solution to the diffusion equation which significantly simplified the necessary code and is the reason we chose an explicit method for our final codes.

3.3 Examples of Reaction-Diffusion Equations in One Dimension

The behavior of reaction diffusion equations depends strongly on the reaction term so it is difficult to talk about their behavior generally. Instead we present some examples of interesting observed behavior which appeared for different reaction terms and initial conditions. In all of the presented examples we used the explicit formulation of the reaction term (Adams-Bashforth) which we outlined above.

3.3.1 Example 4: Dirichlet Boundary Conditions with $f(u) = u(1-u)$

A first example of interesting behavior introduced by a reaction term comes from using the reaction term $f(u) = u(1-u)$ in the setup of example 1. Recall example 1 used Dirichlet boundary conditions on [0,1] with the initial condition $g(x) = sin(6\pi x) + x - 1$. The early behavior of the solution does not appear significantly different that that of the diffusion equation but we see as time goes on rather than converging to a steady state the solution goes to minus infinity. Below are plots of the behavior, both for a number of discrete times, and continuously via a color plot. For times greater than around 1180 the solution becomes too small (values approaching minus infinity) so the maximum time shown is 1000.

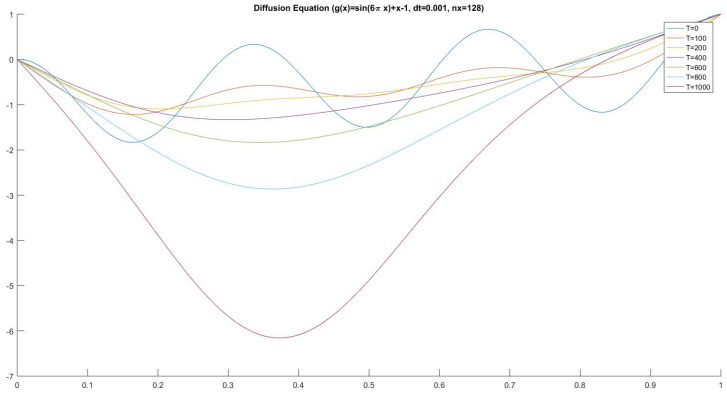

Figure 13: Long term solution behavior of example 5

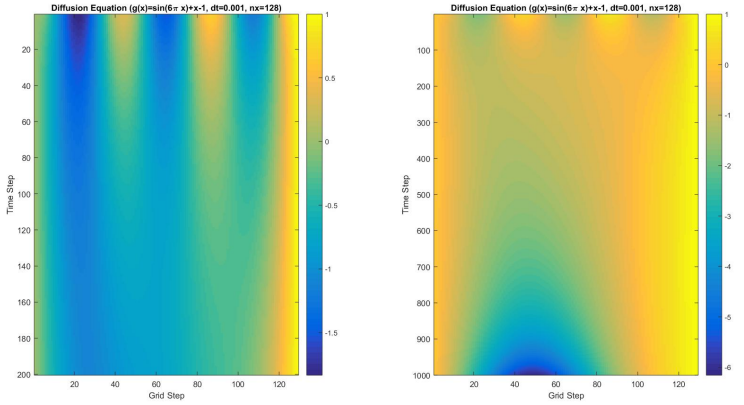

Figure 14: Long term solution behavior of example 5

Using the same methodology we used for the diffusion equation we checked the order of accuracy of our method with the introduced reaction term and as in the diffusion equation we found it to be approximately order h^2 with the ratio of errors approximately twice the ratio of interval sizes.

```
f =

  Columns 1 through 10

      NaN    1.4262    5.0158    7.6600    3.4292    4.7432    6.5471    4.0659    7.3213    3.7315
      NaN    5.1492    3.9549    5.7378    7.1444    5.2552    5.2255    5.1846    5.1372    4.9810
      NaN    3.9823    4.4028    4.5540    6.4516    4.6399    4.6286    4.6639    4.5501    4.5969
      NaN    5.0836    5.0770    5.1659    6.0384    5.1987    5.1959    5.2173    5.1711    5.1882
      NaN       Inf       Inf       Inf       Inf       Inf       Inf       Inf       Inf       Inf

  Columns 11 through 17

    9.0539    4.0425    0.8711    8.6985    5.1654    2.4365       NaN
    6.5145    6.1488    5.6223    5.3933    4.0989    5.7308       NaN
    5.1687    4.9471    4.7877    4.4986    4.5322    3.9753       NaN
    5.4386    5.3440    5.2389    5.1575    5.1193    5.1094       NaN
       Inf       Inf       Inf       Inf       Inf       Inf       NaN
```

Figure 15: Relative error for example 5

Using the same reaction term, boundary conditions, and initial condition but extending the domain to [0,5] resulted in the same behavior. The solution went to minus infinity at approximatly the same value of x, slightly less than 0.5.

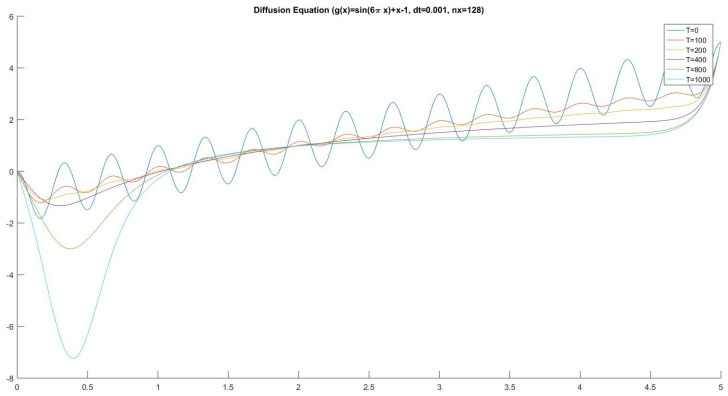

Figure 16: Long term solution behavior of example 4 on the extended domain [0,5]

3.3.2 Example 5: Dirichlet Boundary Conditions with $f(u) = u(1 - u^2)$

A second example comes from repeating example 4 with the new reaction term $f(u) = u(1 - u^2)$. Unlike in example 5 the solution tended towards a finite steady state and the solution depended on the size of the interval considered. Below are plots of the solution over time with $\Delta x = 1/256$ for the reaction-diffusion equation on [0,1], [0,5], and [0,10]. The steady state of the solution on [0,1] is unique while the solutions on [0,5] and [0,10] appear to behave similarly with a dip on [0,1] followed by stabilizing around 1 and then jumping up to meet the boundary condition at the end of the interval. I checked other intervals ([0,3] and [0,7]) which are not pictured and observed the same behavior.

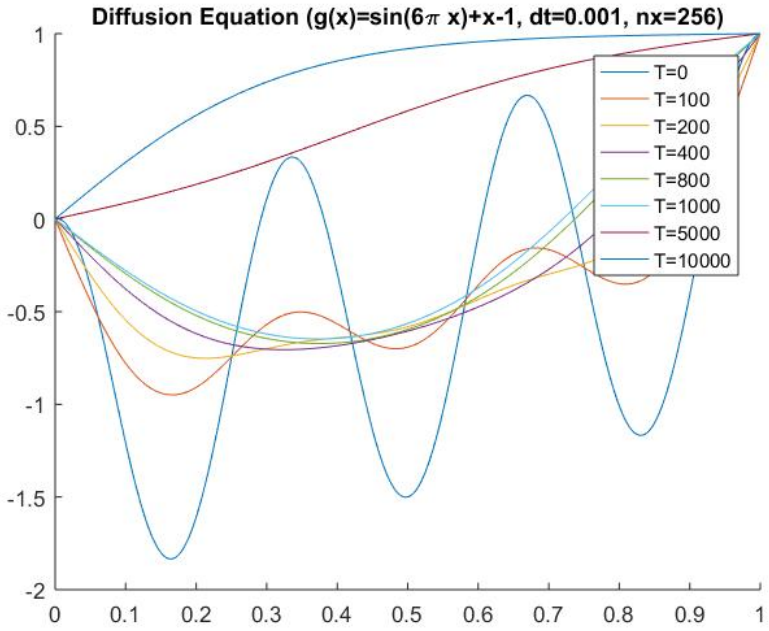

Figure 17: Long term solution behavior on [0,1]

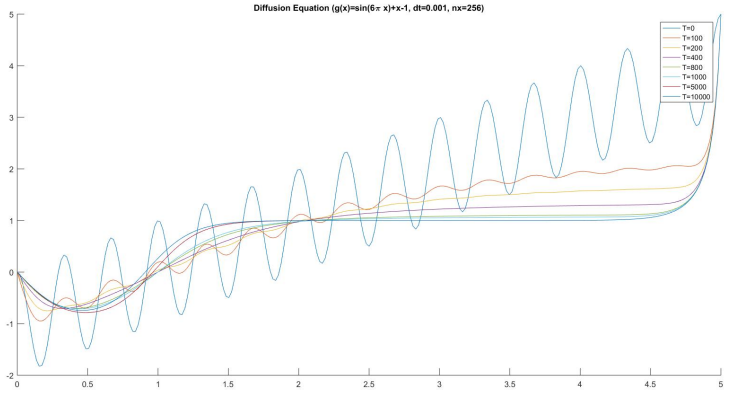

Figure 18: Long term solution behavior on [0,5]

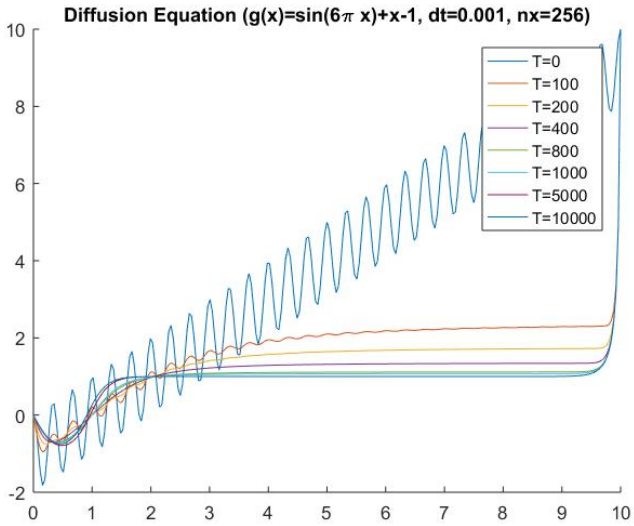

Figure 19: Long term solution behavior on [0,10]

Another reaction term is $f(u) = u(1-u)(u-a)$ where $a \in [0,1]$. The behavior of the reaction diffusion equation with this term appeared to be similar to the behavior of the reaction diffusion equation with the reaction term $f(u) = u(1-u^2)$. Pictured below are plots of the long term behavior of $a = 0.25$ and $a = 0.75$ on the domain [0,5].

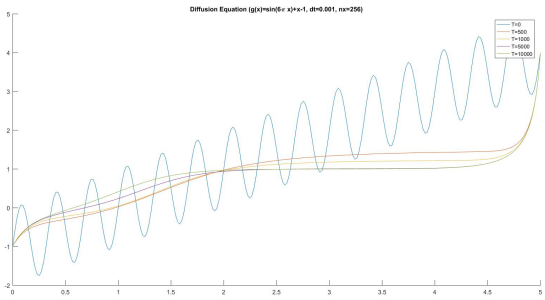

Figure 20: Long term behavior with $f(u) = u(1-u)(u-0.25)$ on [0,5]

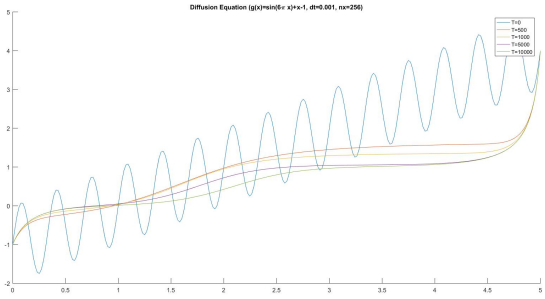

Figure 21: Long term behavior with $f(u) = u(1-u)(u-0.75)$ on [0,5]

3.3.3 Example 6: Neumann Boundary Conditions with $f(u) = u(1-u)$

Here we repeat the initial and boundary conditions of Example 2 with the addition of the reaction term $f(u) = u(1-u)$. Recall the initial condition was $g(x) = 3x + 3\sin(6\pi x)$ and the boundary conditions were Neumann. We see that the solution initially tends towards becoming linear on the middle of the domain but at the boundaries the solution tends towards minus infinity on the left and plus infinity on the right. This behavior appeared to be consistent regardless of the size of the domain. See the two plots below, the code needed to be stopped early because the values at the boundaries and subsequently throughout the domain grew large very quickly.

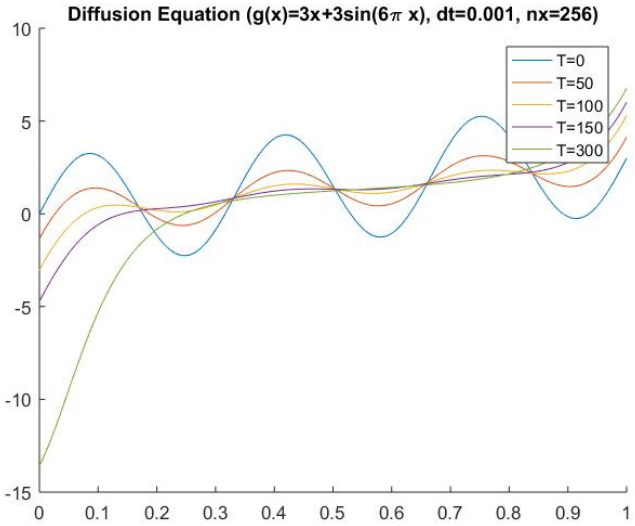

Figure 22: Long term solution behavior on [0,1] with $f(u) = u(1-u)$

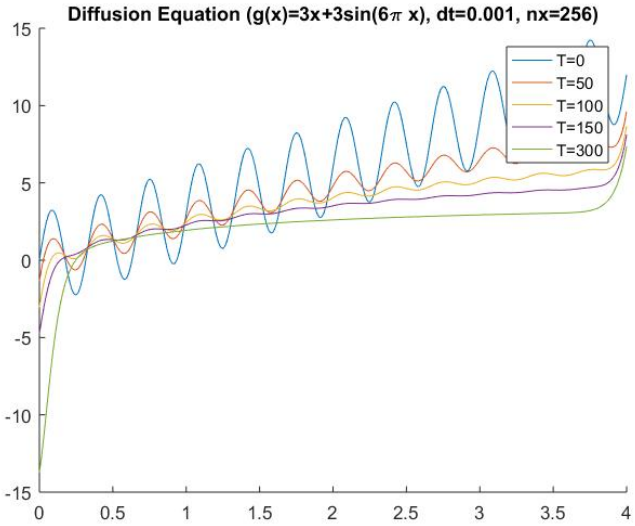

Figure 23: Long term solution behavior on [0,4] with $f(u) = u(1-u)$

3.3.4 Example 7: Periodic Boundary Conditions with $f(u) = u(1-u)$

Here we repeat the initial and boundary conditions of example 3 with the addition of the reaction term $f(u) = u(1-u)$. Recall the initial condition was $g(x) = 2\cos(4\pi x) + 4\sin(6\pi x)$ and the boundary conditions were periodic. We see that the solution initially appears similar to example 3 but as time goes on it tends towards a periodic solution which goes towards minus infinity. This is clearer on the larger domain [0,4] where the form of the solution is clearer. The code needed to be stopped early because values grew large in absolute value very quickly.

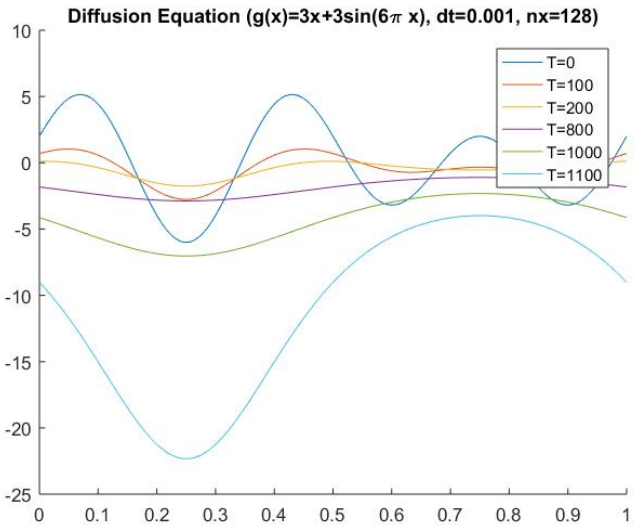

Figure 24: Long term solution behavior on [0,1] with $f(u) = u(1-u)$

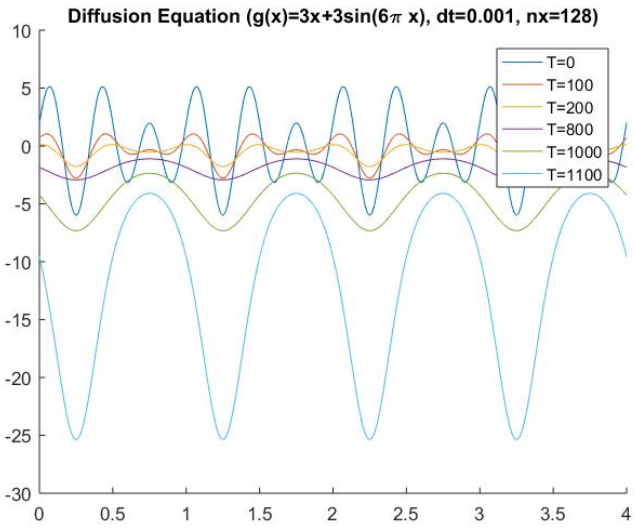

Figure 25: Long term solution behavior on [0,4] with $f(u) = u(1-u)$

3.3.5 Example 8: Dirichlet Boundary Conditions with $f(x,t) = cos(2\pi t)(erf(12(x-a)) - erf(12(x-b)))$

A third example comes from using the initial conditon $u(x,0) = 0$ for all x in the domain with a reaction term $f(x,t) = cos(2\pi t)(erf(12(x-a)) - erf(12(x-b)))$ where a and b are constants. Considering different domains and adjusting a and b it appears that solution has zeros at a and b with something like a single sine wave centered at a and b which connect smoothly in the middle of the domain.

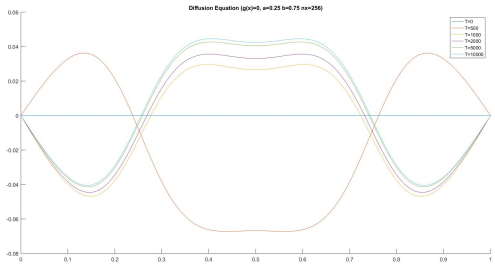

Figure 26: Long term solution behavior on [0,1] with $a = 0.25$ and $b = 0.75$

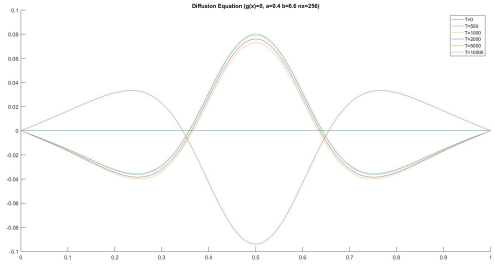

Figure 27: Long term solution behavior on [0,1] with $a = 0.40$ and $b = 0.60$

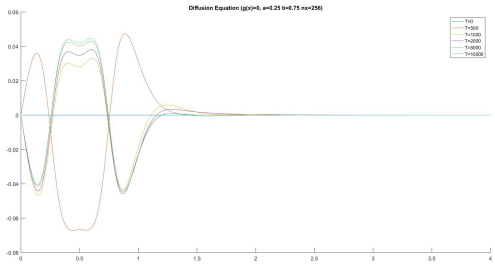

Figure 28: Long term solution behavior on [0,4] with $a = 0.25$ and $b = 0.75$

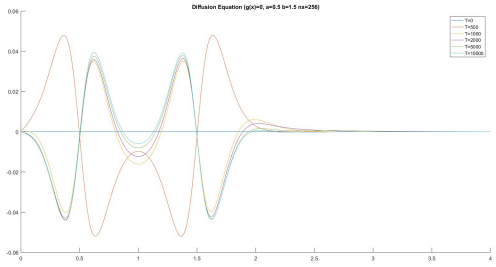

Figure 29: Long term solution behavior on [0,4] with $a = 0.50$ and $b = 1.50$

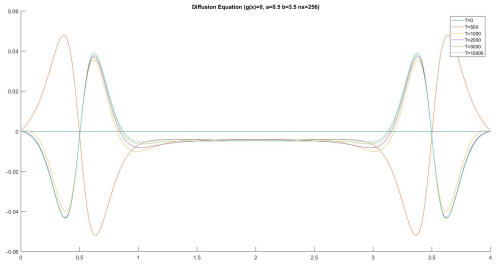

Figure 30: Long term solution behavior on [0,4] with $a = 0.50$ and $b = 3.50$

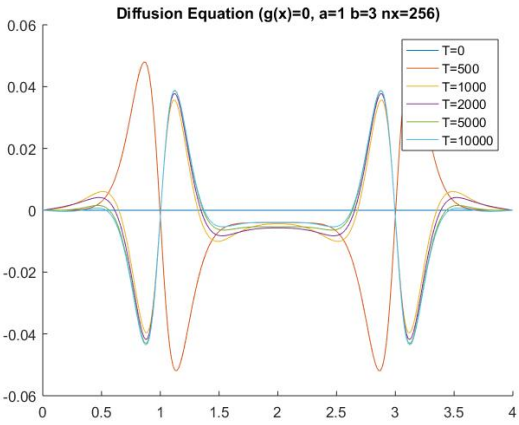

Figure 31: Long term solution behavior on [0,4] with $a = 1$ and $b = 3$

3.3.6 Example 9: Neumann Boundary Conditions with $f(x,t) = cos(2\pi t)(erf(12(x-a)) - erf(12(x-b)))$

A fourth example of a reaction diffusion equation comes from the initial condition $u(x,0) = 0$ with the reaction term $f(x,t) = cos(2\pi t)(erf(12(x-a)) - erf(12(x-b)))$ and Neumann Boundary Conditions. The solution appears to be almost identical to the same equation modeled with Dirichlet conditions. The chief difference is that the solution does not always go to zero at the boundary of the domain.

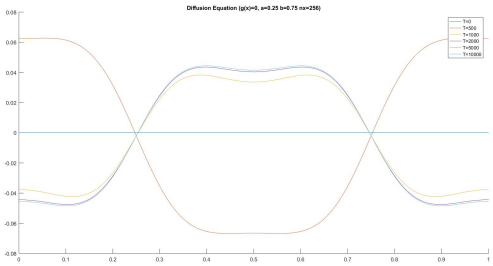

Figure 32: Long term solution behavior on [0,1] with $a = 0.25$ and $b = 0.75$

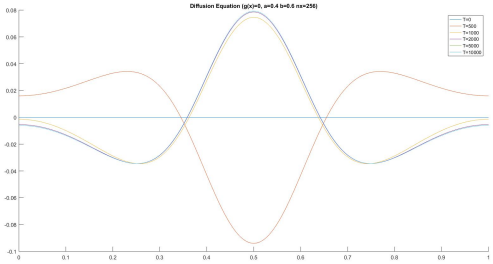

Figure 33: Long term solution behavior on [0,1] with $a = 0.40$ and $b = 0.60$

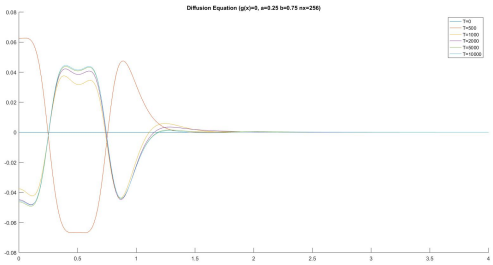

Figure 34: Long term solution behavior on [0,4] with $a = 0.25$ and $b = 0.75$

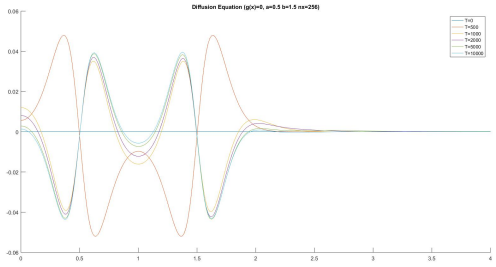

Figure 35: Long term solution behavior on [0,4] with $a = 0.50$ and $b = 1.50$

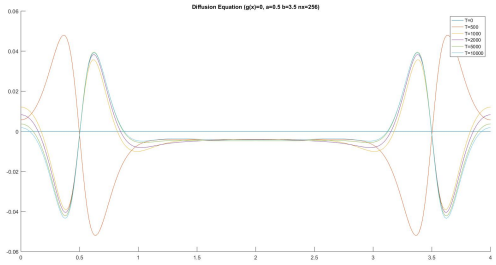

Figure 36: Long term solution behavior on [0,4] with $a = 0.50$ and $b = 3.50$

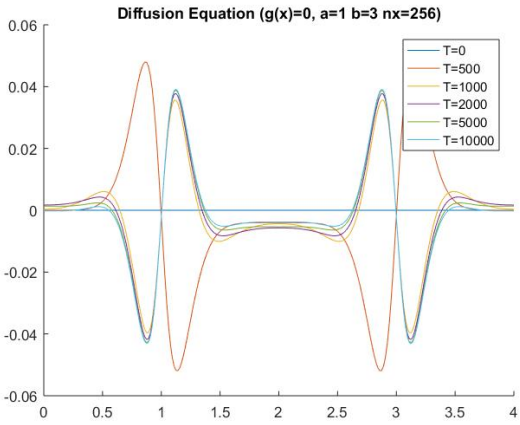

Figure 37: Long term solution behavior on [0,4] with $a = 1$ and $b = 3$

3.3.7 Example 10: Periodic Boundary Conditions with $f(x,t) = cos(2\pi t)(erf(12(x-a)) - erf(12(x-b)))$

Here we again use the reaction term $f(x,t) = cos(2\pi t)(erf(12(x-a)) - erf(12(x-b)))$ but with periodic boundary conditions. The behavior appears almost identical to the behavior of the same equation with Neumann boundary conditions (Example 8). Below are examples of the solution for different domains and values of a and b.

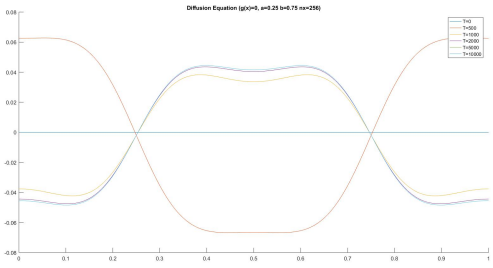

Figure 38: Long term solution behavior on [0,1] with $a = 0.25$ and $b = 0.75$

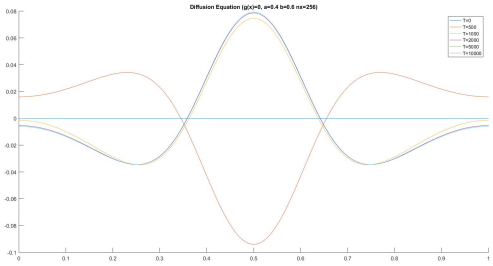

Figure 39: Long term solution behavior on [0,1] with $a = 0.40$ and $b = 0.60$

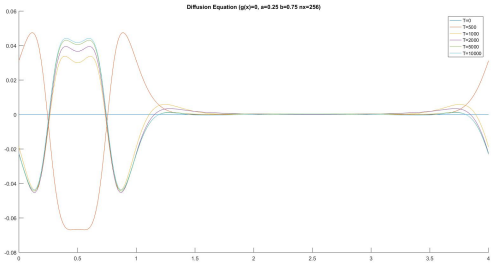

Figure 40: Long term solution behavior on [0,4] with $a = 0.25$ and $b = 0.75$

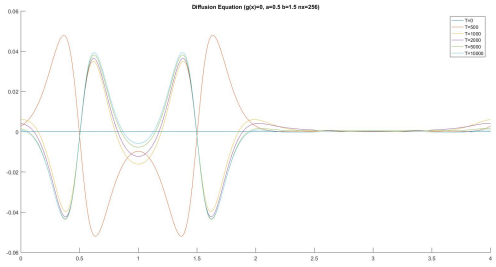

Figure 41: Long term solution behavior on [0,4] with $a = 0.50$ and $b = 1.50$

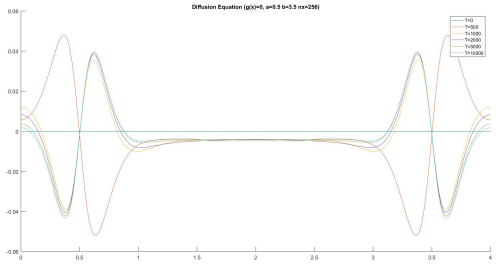

Figure 42: Long term solution behavior on [0,4] with $a = 0.50$ and $b = 3.50$

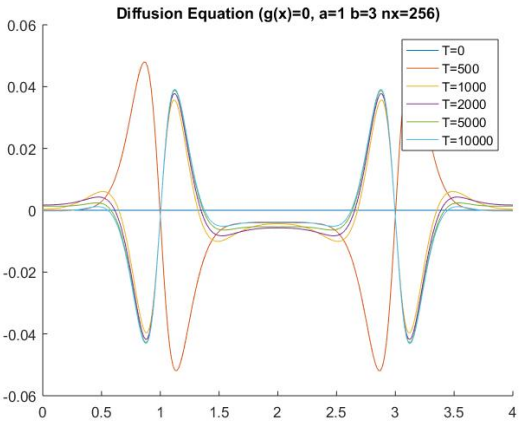

Figure 43: Long term solution behavior on [0,4] with $a = 1$ and $b = 3$

4 Diffusion Equation in Two Dimensions

In two dimensions the diffusion equation is given by:

$$\frac{\partial u}{\partial t} = \mu \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) \tag{6}$$

The Crank-Nicolson method may be extended to two dimensions but the resulting system is no longer tridiagonal so it is not as efficient as in one dimension. Instead we implemented the alternating direction implicit (ADI) method to solve the two dimensional diffusion equation. The ADI method uses two steps, the first models x implicitly and y explicitly and the second models y implicitly and x explicitly each over half of a time step. Although each step is only first order "because of the symmetry of the two steps, however, the local error introduced in the second step almost exactly cancels out the local error introduced in the first step, so that the combined method is second order accurate over the full timestep" (LeVeque, 168). Furthermore since U^* approximates the solution at time $n+1/2$ it is possible to simply evaluate the given boundary conditions at the intermediate timestep while maintaining second-order accuracy.

The first step of the method is given by:

$$\left(I - \frac{\Delta t}{2} D_x^2 \right) U^* = \left(I + \frac{\Delta t}{2} D_y^2 \right) U^n$$
$$\Rightarrow \qquad (1+2r)u_{i,j}^* - r(u_{i+1,j}^* + u_{i-1,j}^*) = (1-2r)u_{i,j}^n + r(u_{i,j-1}^n + u_{i,j+1}^n)$$

The second step of the method is given by:

$$\left(I - \frac{\Delta t}{2} D_y^2 \right) U^{n+1} = \left(I + \frac{\Delta t}{2} D_x^2 \right) U^*$$
$$\Rightarrow \qquad (1+2r)u_{i,j}^{n+1} - r(u_{i,j+1}^{n+1} + u_{i,j-1}^{n+1}) = (1-2r)u_{i,j}^* + r(u_{i-1,j}^* + u_{i+1,j}^*)$$

At each timestep the system is block tridiagonal in the case of Dirichlet or Neumann boundary conditions so the Thomas algorithm is an effective means of solving the systems. There will be $2n$ tridiagonal systems to solve for each timestep with n appearing on each of the two steps. In the case of periodic boundary conditions the number of systems to be solved is the same but because the coefficient matrices are no longer tridiagonal an LU decomposition or other means of solving linear systems needs to be employed.

4.1 Example 11: Diffusion Equation in Two Dimensions

Here we only provide a single example of the diffusion equation in two dimensions since additional examples will be given when we treat the reaction diffusion equation in two dimensions. For this example we used the initial condition $g(x, y) = \sin(4\pi x) + 2x\cos(2\pi y)$ on the domain $[0, 1] \times [0, 1]$ with homogeneous Neumann boundary conditions. Our timestep size for this example was 0.001. Below is a plot of the evolution of the solution over time.

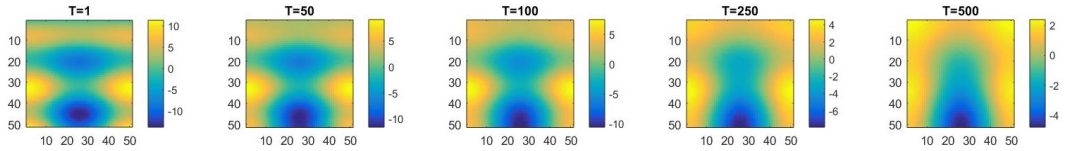

Figure 44: Evolution of the solution of example 11 over time

We calculated the error following the same methodology used in one dimension - we calculated the value of the solution at time t=0.5 (the final time shown in the figure above) using $h(= dx = dy = dt)$ values of 1/16, 1/32, 1/64, 1/128, and 1/256 and treated the obtained solution using $h = 1/256$ as the true solution. We then computed the ratio of successive errors and averaged that ratio over all x and y values at each timestep. This produced the numbers shown below all exceeding 4 which confirm our method was second order.

```
favg =

    4.0840    4.1918    4.9836
```

Figure 45: Ratio of error between successive h values

5 Reaction-Diffusion Equation in Two Dimensions

In two dimensions the reaction-diffusion equation is given by

$$\frac{\partial u}{\partial t} = \mu \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) + f(t, x, u) \tag{7}$$

As in the one-dimensional case it is possible to model the reaction term either implicitly or explicitly. As in the one-dimensional case the system becomes nonlinear when modeling the reaction term implicitly but remain linear when modeling the reaction term explicitly. In the one-dimensional case we implemented both implicit and explicit methods but in two dimensions we only implemented an explicit method. We again used the Adams-Bashforth two step method to model the nonlinear term together with the ADI method. This led to our two steps being:

$$(1+2r)u^*_{i,j} - r(u^*_{i+1,j} + u^*_{i-1,j}) = (1-2r)u^n_{i,j} + r(u^n_{i,j-1} + u^n_{i,j+1}) + \frac{\Delta t}{2}(3f(u^n_{i,j}) - f(u^{n-1}_{i,j}))$$

$$(1+2r)u^{n+1}_{i,j} - r(u^{n+1}_{i,j+1} + u^{n+1}_{i,j-1}) = (1-2r)u^*_{i,j} + r(u^*_{i-1,j} + u^*_{i+1,j}) + \frac{\Delta t}{2}(3f(u^n_{i,j}) - f(u^{n-1}_{i,j}))$$

which forms a tridiagonal system at each step of the ADI method which can be solved efficiently using the Thomas Algorithm except in the case of periodic boundary conditions for which an LU factorization followed by solving the upper and lower triangular systems can be used instead.

5.1 Two Dimensional Reaction-Diffusion Examples

As in the one dimensional case the behavior of a two dimensional reaction diffusion systems depends strongly on the boundary conditions, reaction term, and initial conditions. Below are a number of examples of 2D reaction diffusion systems both with and without reaction terms.

5.1.1 Example 12: Neumann Boundary Conditions with $f(u) = u(1-u)$

A first example comes from using homogeneous Neumann boundary conditions with a random initial condition (the initial condition was randomized in blocks so each 2 by 2 block was initialized to the same value) on a 50 by 50 grid on $[0,1] \times [0,1]$. The 2D Neumann boundary condition code was then run for 1000 timesteps with a timestep size of 0.001. Below are four sets of images, the first corresponds to the case of no reaction term with the initialized values in [0,1] and the second corresponds to the same initial condition with the reaction term $f(u) = u(1-u)$. Comparing the two it does not appear that the reaction term had a significant impact on the evolution of the solution but it does appear to produce some differences in the solution. It appears to add some symmetry. We might expect the impact to be small since $u(1-u)$ will be small.

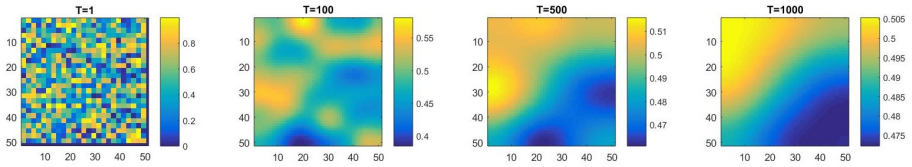

Figure 46: Solution over time with random [0,1] initial condition and no reaction term

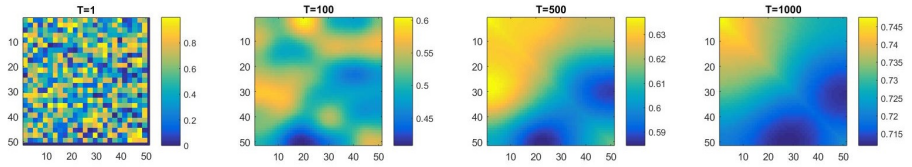

Figure 47: Solution over time with random [0,1] initial condition and reaction term $f(u) = u(1-u)$

The next two figures show how the solution evolves when the initial condition is a random number in [0,5] without and with the reaction term $f(u) = u(1-u)$. In this case the reaction term appears to have a more significant impact and appears to produce symmetry in the solution.

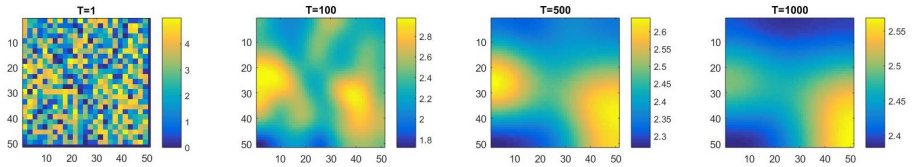

Figure 48: Solution over time with random [0,1] initial condition and no reaction term

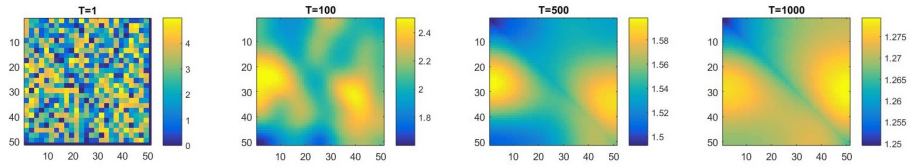

Figure 49: Solution over time with random [0,1] initial condition and reaction term $f(u) = u(1-u)$

5.1.2 Example 13: Periodic Boundary Conditions with $f(u) = u(1-u)$

A second example comes from using periodic boundary conditions with a random initial condition (the initial condition was randomized in blocks so each 2 by 2 block was initialized to the same value in [0,10], afterwards the values at the corners and bottom and left edges were adjusted so the initial condition was periodic) on a 50 by 50 grid on $[0,1] \times [0,1]$. The 2D periodic code was then run for 1000 timesteps with a timestep size of 0.001. Below are two sets of images, the first corresponding to running the code with no reaction term and the second and third corresponding to running the code with the reaction terms $f(u) = u(1-u)$ and $f(u) = u(1-u^2)$ from the same initial condition. Comparing the three it is clear that the reaction term had a significant impact on the evolution of the solution over time and that the different reaction terms resulted in a difference in behavior.

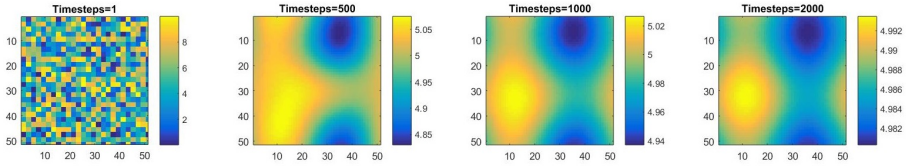

Figure 50: Solution over time with random [0,10] initial condition and no reaction term

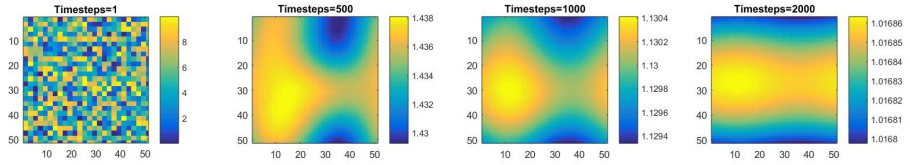

Figure 51: Solution over time with random [0,10] initial condition and reaction term $f(u) = u(1-u)$

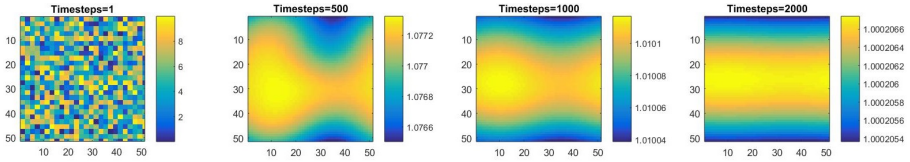

Figure 52: Solution over time with random [0,10] initial condition and reaction term $f(u) = u(1-u^2)$

6 Gray-Scott Model

The Gray-Scott equations model the interactions of two chemical species by reaction and diffusion. Under the right conditions these reactions can produce interesting patterns formed by plotting the concentrations of one of the chemical species. The Gray Scott equations are given by:

$$\frac{\partial u}{\partial t} = \nu_u \Delta u - uv^2 + f(1-u) = \nu_u \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) - uv^2 + f(1-u) \tag{8}$$

$$\frac{\partial v}{\partial t} = \nu_v \Delta v + uv^2 - (f+k)v = \nu_v \left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right) + uv^2 - (f+k)v \tag{9}$$

Here u and v are the two chemical species with diffusivities ν_u and ν_v respectively, a third species p is formed by reaction of u and v. The terms f and k represent the rate of the process which feeds u and drains u, v, and p and the rate of the conversion of v to p.

To model the Gray-Scott equations we implemented (with your help) a finite difference method with Neumann boundary conditions, treating u and v explicitly at each time step and solving the equations for u and v separately. Our method is given by:

$$u_{i,j}^{n+1} = u_{i,j}^n + \nu_u \Delta t \left(\frac{u_{i+1,j}^n - 2u_{i,j}^n + u_{i-1,j}^n}{(\Delta x)^2} + \frac{u_{i,j+1}^n - 2u_{i,j}^n + u_{i,j-1}^n}{(\Delta x)^2} \right) + \Delta t \left(-u_{i,j} v_{i,j}^2 + f(1-u_{i,j}) \right) \tag{10}$$

$$v_{i,j}^{n+1} = v_{i,j}^n + \nu_v \Delta t \left(\frac{v_{i+1,j}^n - 2v_{i,j}^n + v_{i-1,j}^n}{(\Delta x)^2} + \frac{v_{i,j+1}^n - 2v_{i,j}^n + v_{i,j-1}^n}{(\Delta x)^2} \right) + \Delta t \left(u_{i,j} v_{i,j}^2 + (f+k) v_{i,j} \right) \tag{11}$$

Using this method we were able to generate a number of interesting patterns. The pattern generation was extremely sensitive to initial conditions and parameter values. The next section contains a number of the more interesting patterns we were able to produce and the conditions which generated them.

6.1 Gray-Scott Examples

All of the examples shown used $\nu_u = 0.01$ and $\nu_v = 0.005$. The domain considered was $[0, 15] \times [0, 15]$ and was discretized in space as a 151×151 grid. The first four figures shown used $k = 0.060$ with varying values of f and a timestep size of 0.1. It appears that higher values of f led to less connectivity between areas where the concentration of u was high while lower values saw more interconnectedness of areas where the concentration of u was high

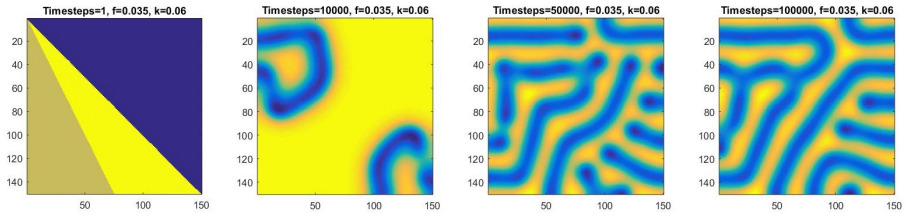

Figure 53: Gray-Scott Model over time with $k = 0.60$ and $f = 0.035$

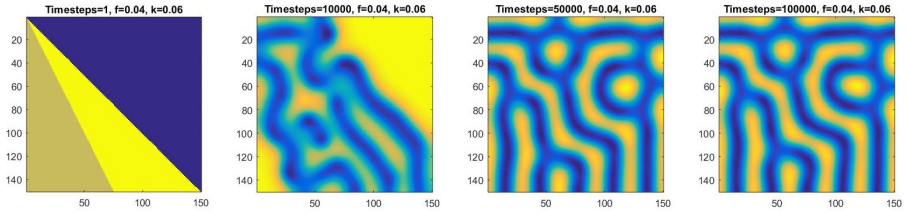

Figure 54: Gray-Scott Model over time with $k = 0.60$ and $f = 0.040$

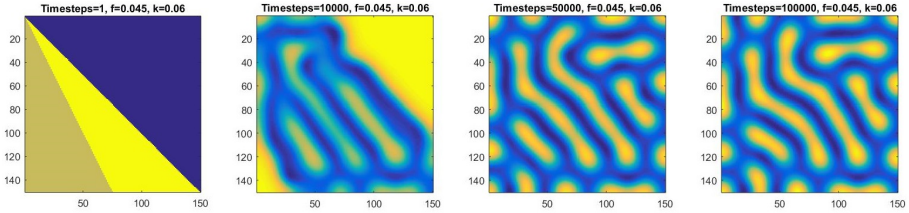

Figure 55: Gray-Scott Model over time with $k = 0.60$ and $f = 0.045$

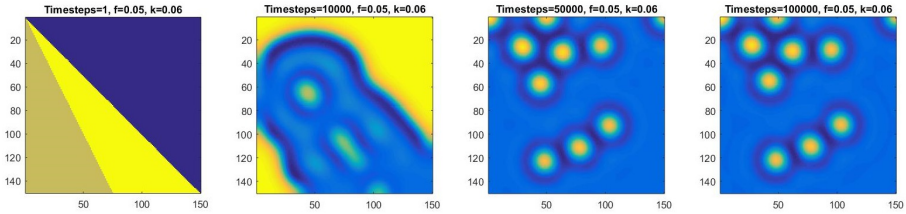

Figure 56: Gray-Scott Model over time with $k = 0.60$ and $f = 0.050$

The next three figures shown used $k = 0.065$ with varying values of f and a timestep size of 0.2. It appears that the same relationship between the value of f and the connectivity of areas of higher concentrations of u exists although it is less pronounced. At higher values of f there were more and more connected regions where the concentration of u was low.

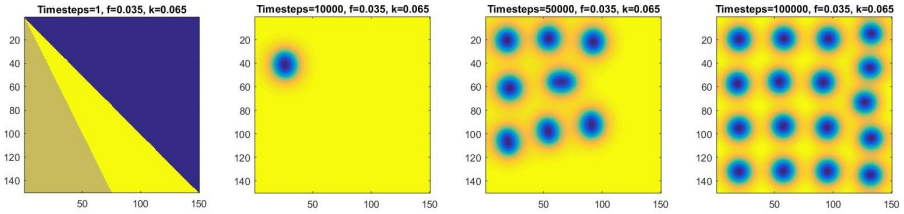

Figure 57: Gray-Scott Model over time with $k = 0.65$ and $f = 0.035$

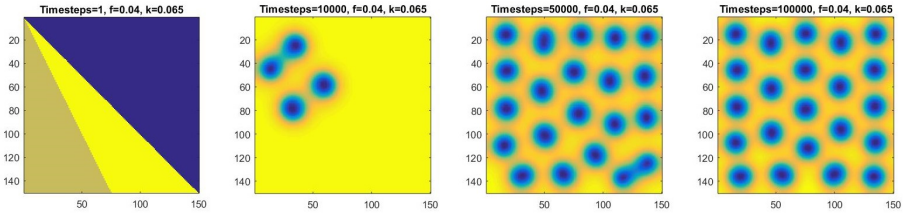

Figure 58: Gray-Scott Model over time with $k = 0.65$ and $f = 0.040$

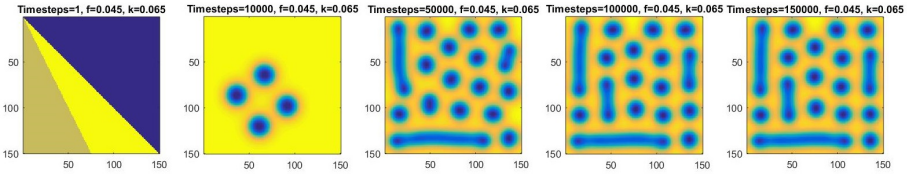

Figure 59: Gray-Scott Model over time with $k = 0.65$ and $f = 0.045$

The final two images used $k = 0.050$ with two values of f and a timestep size of 0.2. The patterns appearing here are more wavelike rather than the more discrete dots or snakes which appeared in the previous two sets of images.

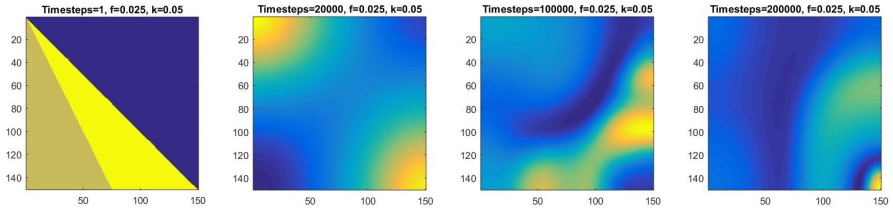

Figure 60: Gray-Scott Model over time with $k = 0.050$ and $f = 0.025$

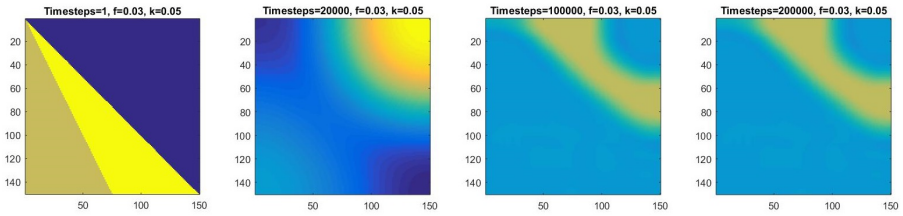

Figure 61: Gray-Scott Model over time with $k = 0.050$ and $f = 0.030$

7 Bibliography

1. LeVeque, Randall J. Finite Difference Methods for Ordinary and Partial Differential Equations: Steady-State and Time-Dependent Problems. SIAM, 2007.

2. Yanyan, Claire, and Flavio Fenton. "Numerical Solutions of Reaction-Diffusion Equations: Application to Neural and Cardiac Models." American Journal of Physics, vol. 84, 2016, doi:https://doi.org/10.1119

3. ADI notes week8, Canvas

4. Abelson, et al. "Gray Scott Model of Reaction Diffusion." Gray Scott Home Page, groups.csail.mit.edu/mac/p

5. Manaa, Saad, and Joli Rasheed. "Successive and Finite Difference Method for Gray Scott Model." Journal of University of Zakho, vol. 1, no. 2, 2013.

6. Chou, Ching-Shan, et al. "Numerical Methods for Stiff Reaction-Diffusion Systems." AIMS, 2007, doi:10.3934/dcdsb.2007.7.515.