5/8/2024

Project 2 Report

MATH 453

Cameron Robinson

Table of Contents

Table of Figures

Project 2

Part 1

The objective of part one was to implement a centered finite difference method for solving Laplace's equation. Figure 1 shows the specific PDE solved in this section, and its exact solution. The code for implementing this method is shown in Figure 30. This method works by creating a matrix (A) containing equations for each unknown point (the interior points), and a vector (b) containing all of the known points (the boundary values). Since there are as many equations as there are unknown values, this matrix system can be directly solved for all the unknown points.

$$Laplace's\ Equation\ (2D)$$

$$u_{xx} + u_{yy} = 0$$

$$u(0,y) = 4y,\ u(1,y) = 4y$$

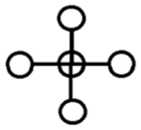
$$u(x,0) = 0,\ u(x,1) = 4$$

$$Solution:$$

$$u(x,y) = 4y$$

Figure 1: PDE for Part 1

Figure 2 illustrates the stencil and update equation for the centered finite difference method. As mentioned, an equation is created for each unknown interior point, u(x,y). Potentially all the points in the stencil are unknown. The only time the values are known is when the stencil overlaps with one of the boundary points. In this case, the x and y step-sizes (h) are equal, however, that does not have to be the case.



$$4u(x,y) - u(x+h,y) - u(x-h,y) - u(x,y+h) - u(x,y+h) = 0$$

Figure 2: Finite Difference Stencil and Update Equation for 2D Laplace's Equation

Figure 3 shows a graph of the numerical solution calculated with 20 steps in the x and y directions. Figure 4 shows the exact solution, and Figure 5 shows graphs of the errors between the exact solutions and numerical solutions. Since the solution to the PDE is a plane, there was no difference between the 5, 10, or 20-step numerical solutions and the exact solution. Although the error graphs seem to indicate that there was some error, this is most likely due to rounding error when calculating the difference between the numerical and exact solutions. This explanation is supported by Figure 6, which shows that there is no difference between the solution matrices for the exact and the 10-step numerical solution.

The centered finite difference method for the wave equation should have error O(delta-x^2) + O(delta-y^2), but since the solution was so simple, the error and convergence of the method are not shown in the solution graphs. The code to create the graphs is shown in Figure 31 and Figure 32.

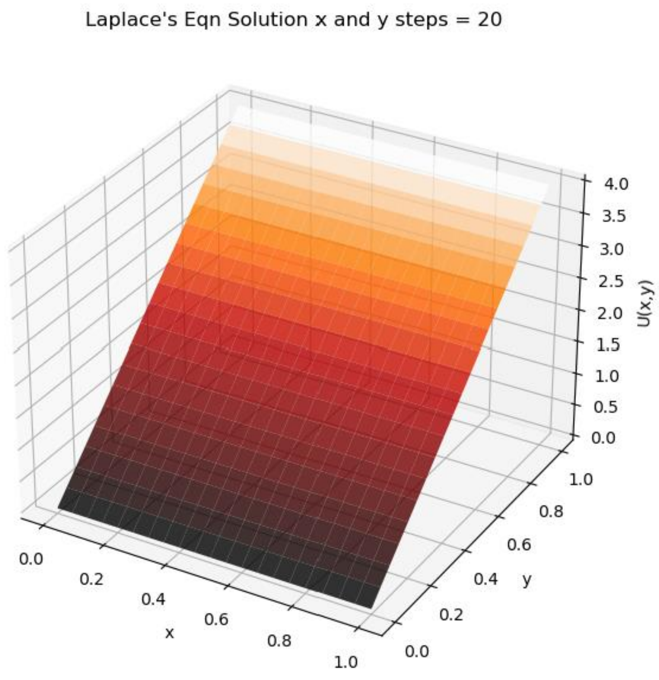


Figure 3: Graph of Numerical Solution with 20 Steps in X and Y

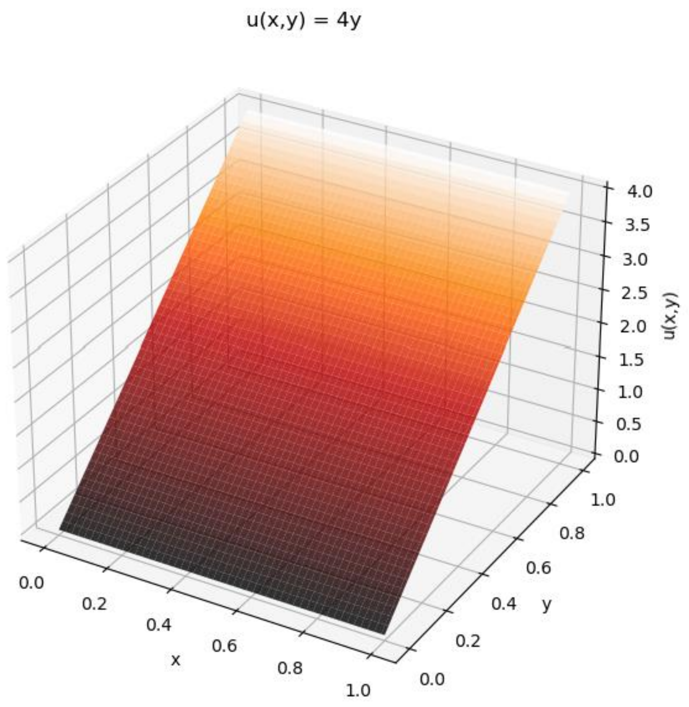


Figure 4: Part 1 Exact Solution Graph

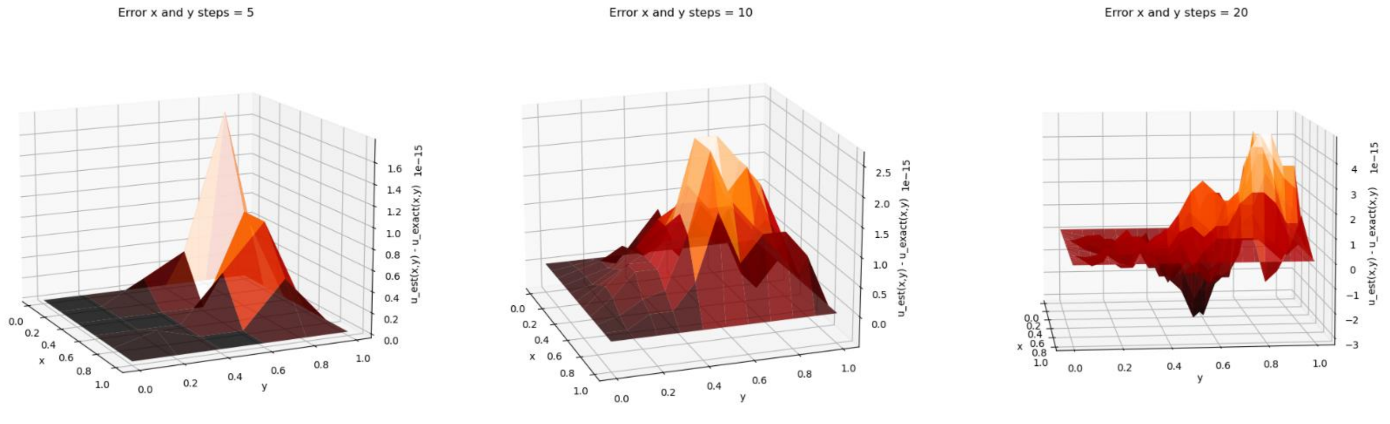
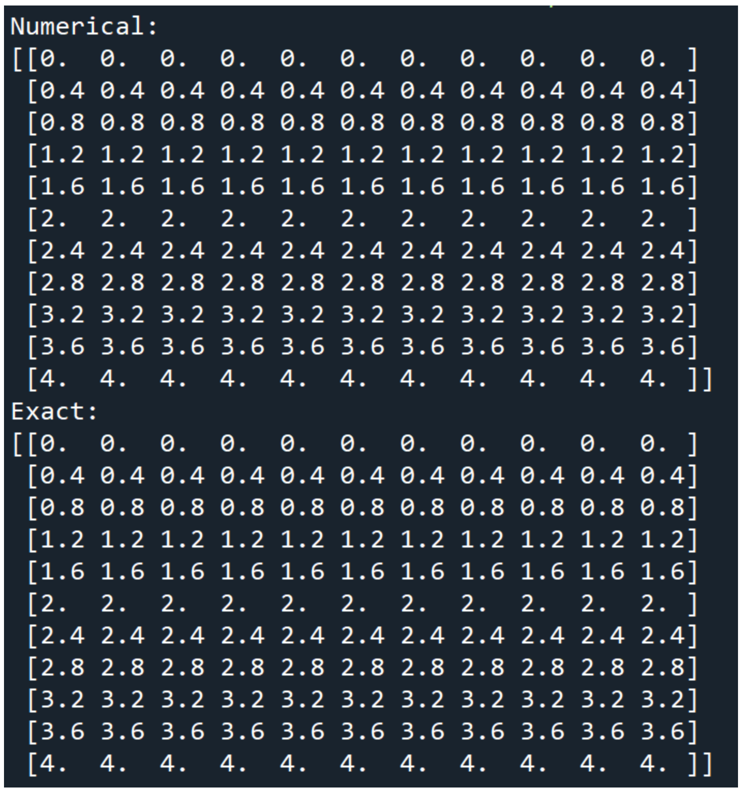


Figure 5: Error Graphs for 5, 10, and 20 Step Solutions



```
Numerical:
[[0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0. ]
 [0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4]
 [0.8 0.8 0.8 0.8 0.8 0.8 0.8 0.8 0.8 0.8 0.8]
 [1.2 1.2 1.2 1.2 1.2 1.2 1.2 1.2 1.2 1.2 1.2]
 [1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6]
 [2.  2.  2.  2.  2.  2.  2.  2.  2.  2.  2. ]
 [2.4 2.4 2.4 2.4 2.4 2.4 2.4 2.4 2.4 2.4 2.4]
 [2.8 2.8 2.8 2.8 2.8 2.8 2.8 2.8 2.8 2.8 2.8]
 [3.2 3.2 3.2 3.2 3.2 3.2 3.2 3.2 3.2 3.2 3.2]
 [3.6 3.6 3.6 3.6 3.6 3.6 3.6 3.6 3.6 3.6 3.6]
 [4.  4.  4.  4.  4.  4.  4.  4.  4.  4.  4. ]]
Exact:
[[0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0. ]
 [0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4]
 [0.8 0.8 0.8 0.8 0.8 0.8 0.8 0.8 0.8 0.8 0.8]
 [1.2 1.2 1.2 1.2 1.2 1.2 1.2 1.2 1.2 1.2 1.2]
 [1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6]
 [2.  2.  2.  2.  2.  2.  2.  2.  2.  2.  2. ]
 [2.4 2.4 2.4 2.4 2.4 2.4 2.4 2.4 2.4 2.4 2.4]
 [2.8 2.8 2.8 2.8 2.8 2.8 2.8 2.8 2.8 2.8 2.8]
 [3.2 3.2 3.2 3.2 3.2 3.2 3.2 3.2 3.2 3.2 3.2]
 [3.6 3.6 3.6 3.6 3.6 3.6 3.6 3.6 3.6 3.6 3.6]
 [4.  4.  4.  4.  4.  4.  4.  4.  4.  4.  4. ]]
```

Figure 6: Solution Matrices for Part 1

Part 2

The objective of part two was to implement a centered finite difference method to solve the 1D wave equation with c = sqrt(2) from t = 0 seconds to t = 12 seconds. Figure 7 shows the boundary conditions and solution to the PDE. The medium that the wave is travelling on has a length of π units with fixed-ends; the initial position is a sine wave; and the initial velocity of the wave is zero.

$$Wave\ Equation\ (1D)$$

$$u_{tt} = c^2 u_{xx}$$

$$u(0,t) = u(\pi,t) = 0$$

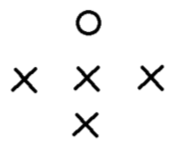
$$u(x,0) = \sin(x),\ u_t(x,0) = 0$$

$$Solution:$$

$$u(x,t) = \cos(ct)\sin(x)$$

Figure 7: Wave Equation PDE for Part 2 and 3

Figure 8 illustrates the stencil and update formula for this method. Unlike Laplace's equation, this method always has four known values and one unknown value, which means that each point can be solved individually. This difference is due to there being two initial conditions, u(x,0) and u_t(x,0) (velocity), which can be used to solve for the position of the wave at the first time-step. The method code can be seen in Figure 33.



$$u(x,t+k) = 2u(x,t) - u(x,t-k) + \frac{k^2c^2}{h^2}(u(x+h,t) - 2u(x,t) + u(x-h,t))$$

Figure 8: 1D Wave Equation Finite Difference Stencil and Update Equation

Figure 9 - Figure 11 are graphs of the numerical solutions for 5, 10, and 20 x-steps respectively. The time step-sizes are calculated using the CFL condition: delta-x/delta-t < c. Since delta-x and c are known, the largest possible time-step can be calculated directly. Figure 12 shows a graph of the actual solution, which looks very similar to the 20-step numerical solution.

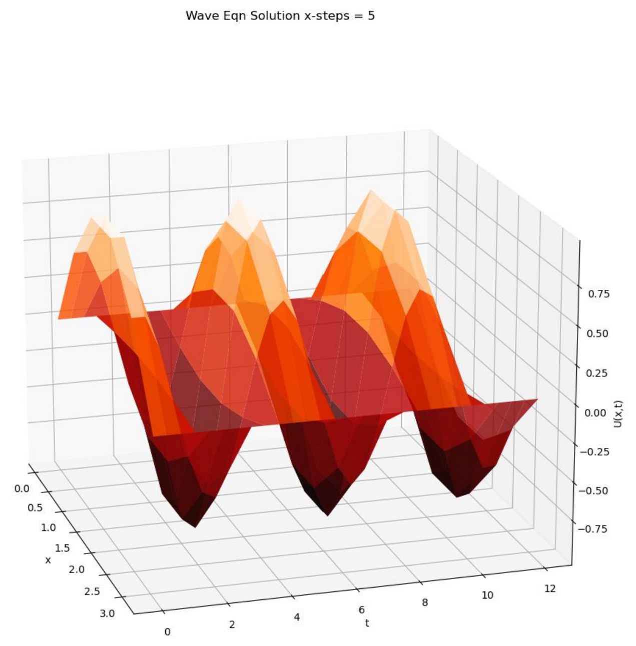


Figure 9: 5-Step Graph of Solution to Part 2

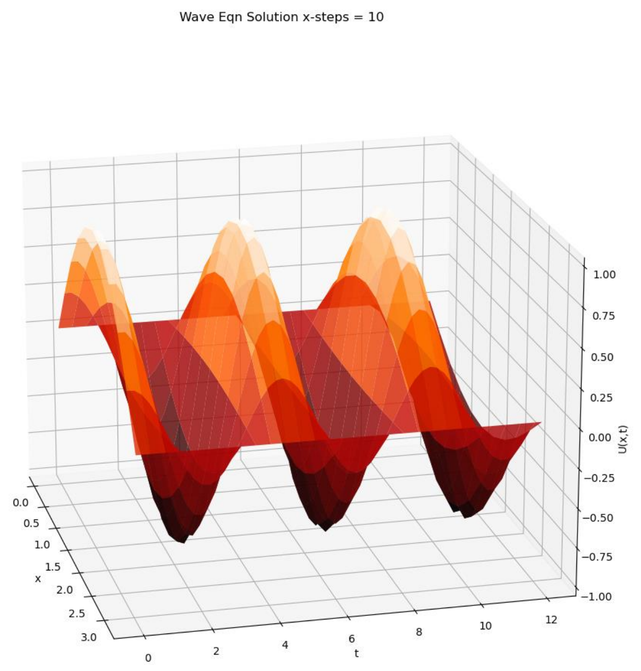


Figure 10: 10-Step Graph of Solution to Part 2

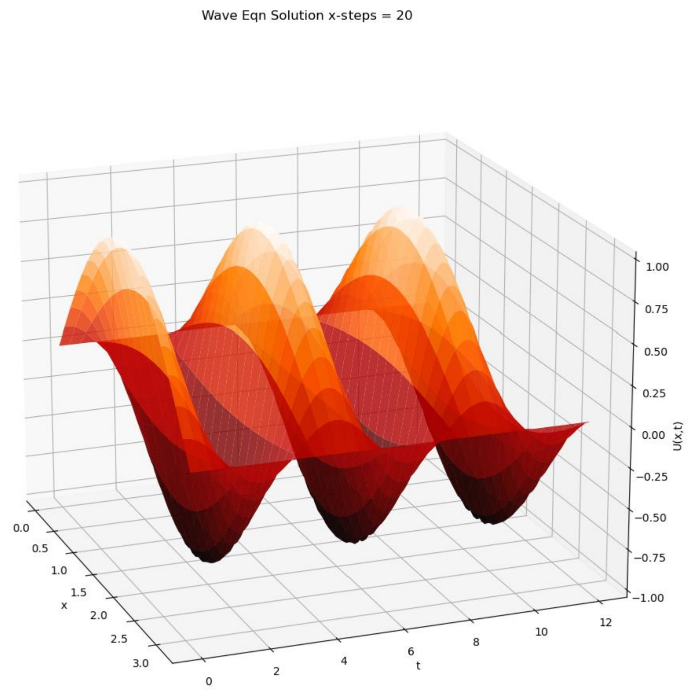


Figure 11: 20-Step Graph of Solution to Part 2

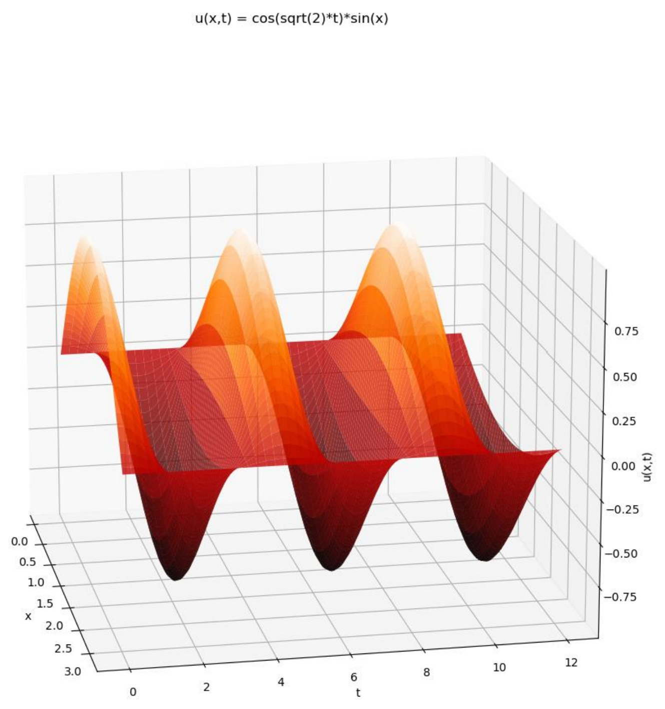


Figure 12: Part 2 Exact Solution

Figure 13 shows the error graphs for the 5, 10, and 20-step numerical solutions. As the number of steps double, the solutions become about two-times more accurate, at least when looking at the maximum error. This is expected since the centered finite difference method has an error term of O(delta-x^2) + O(delta-t^2). The code to create all the graphs in this section can be found in Figure 34.

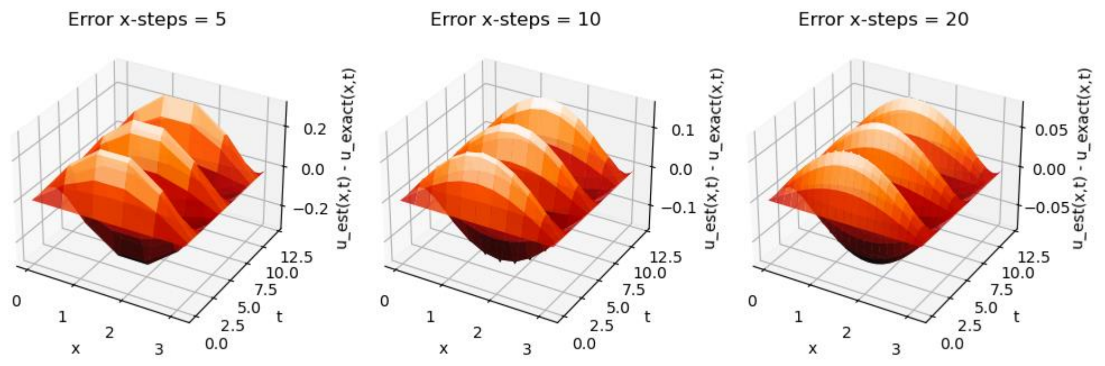


Figure 13: Part 2 Error Graphs

Part 3

The objective of part three was identical to that of part two; the only difference was that the wave speed, c, increased to 3 from sqrt(2). Other than that, the method, PDE, and analytical solution are the same.

Figure 14 - Figure 16 show the numerical solution graphs for 5, 10, and 20 x-steps. Since the only part of the wave that changed from part two was the wave speed increasing, it makes sense that the number of oscillations shown in the solutions have increased. This concept is supported by the analytical solution, where the wave speed modifies the period of cosine, which depends on t (cos(ct)). If c increases, the cosine's period decreases, and the number of oscillations it completes within the same time interval also increases.

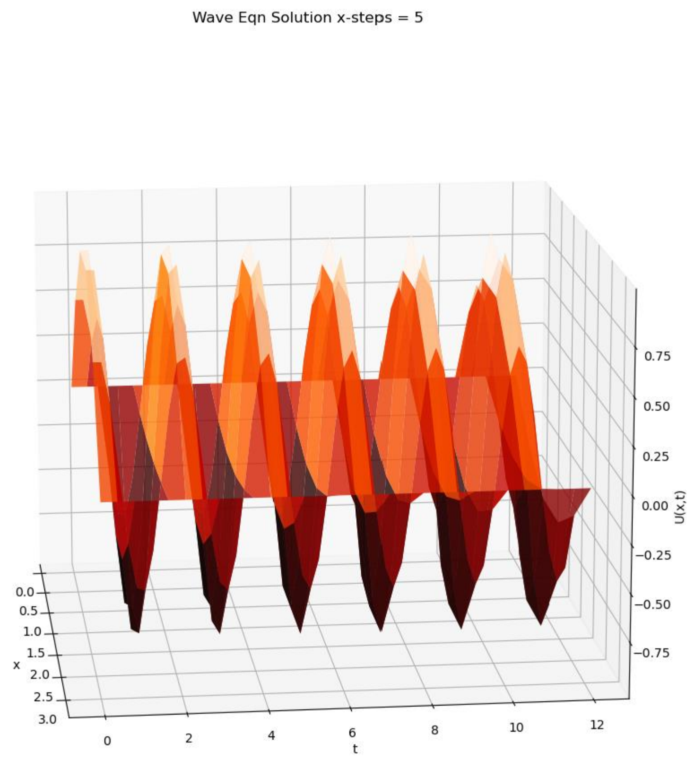


Figure 14: 5-Step Graph of Solution to Part 3

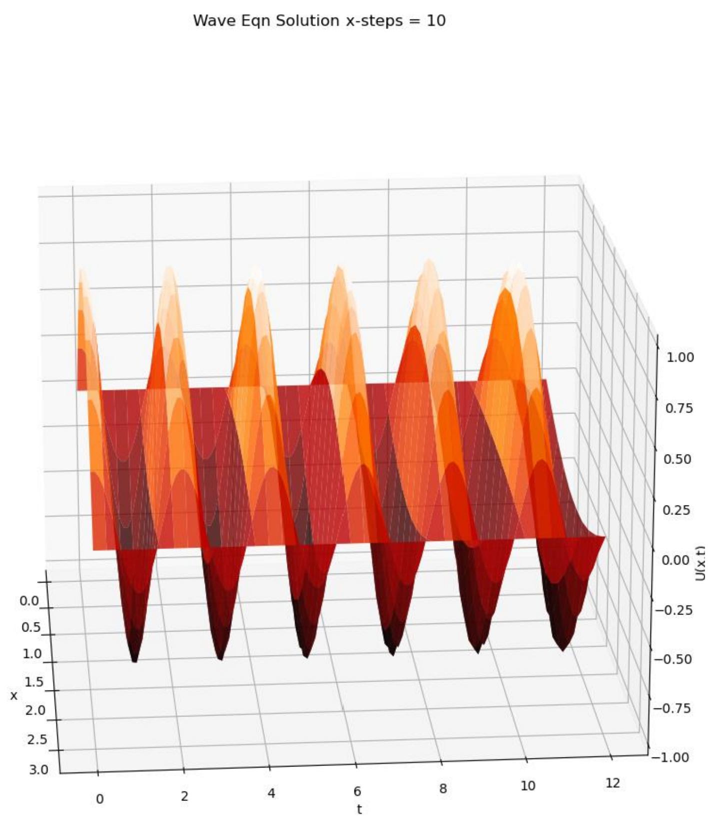


Figure 15: 10-Step Graph of Solution to Part 3

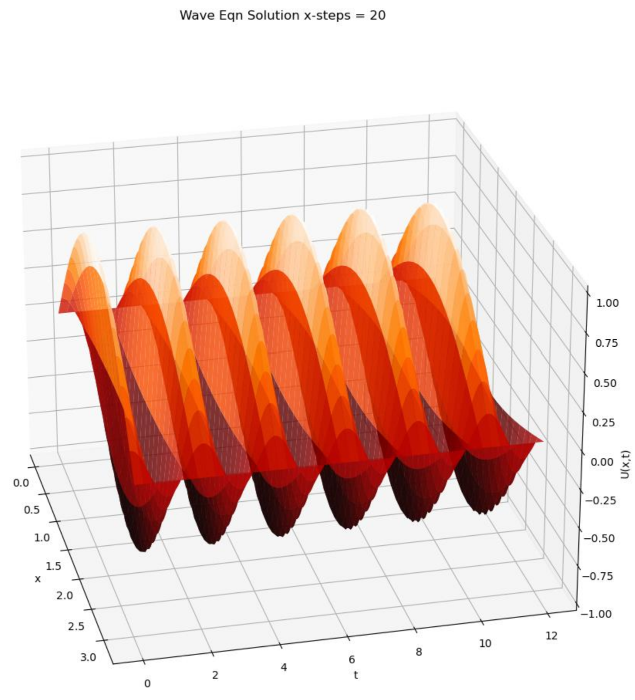


Figure 16: 20-Step Graph of Solution to Part 3

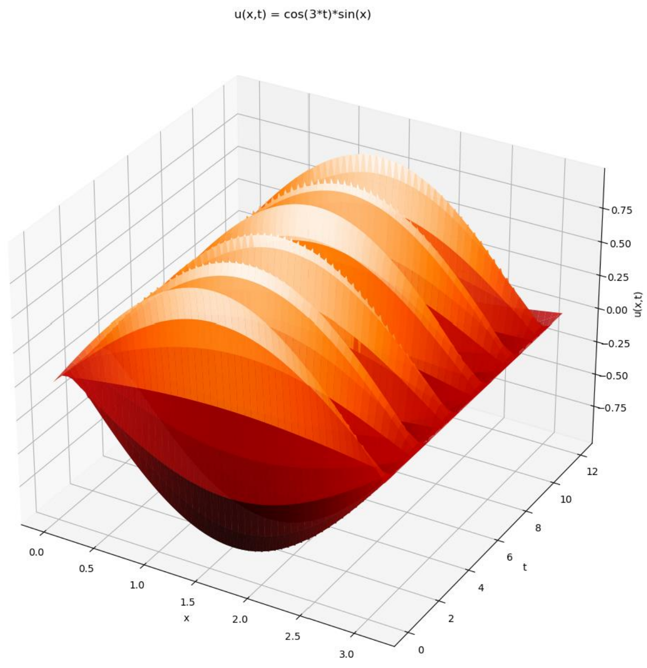


Figure 17: Part 3 Exact Solution

Figure 18 shows the error graphs comparing the above numerical solutions to the exact solution. Like the numerical solutions, the only difference between the error graphs in this section and part two is the number of oscillations. As the number of steps double, the maximum error still decreases by half. It is interesting that the error graphs in part two and three almost perfectly match the shape of the solution graphs. The code to create the graphs in this part is shown in Figure 35.

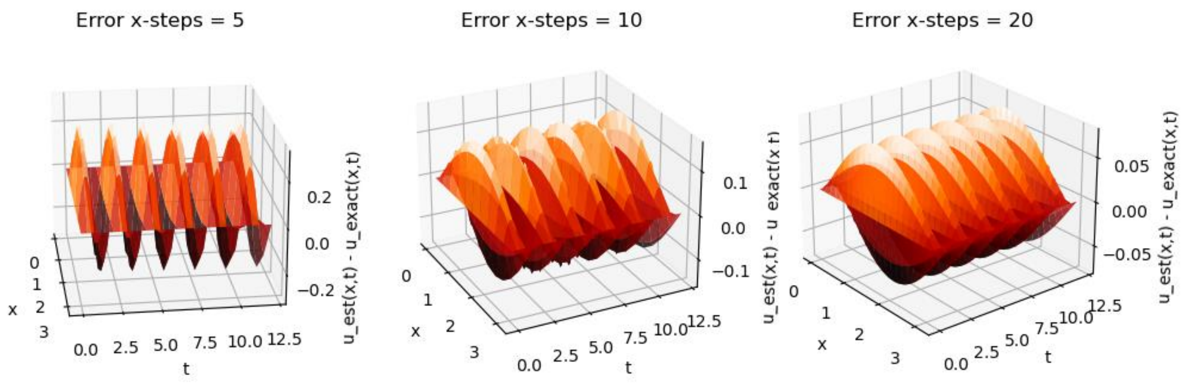


Figure 18: Part 3 Error Graphs

Part 4 and 5

The objective of part four and part five was to solve Laplace's equation for a metal plate with the boundary conditions shown in Figure 19, which also shows analytical solution. This problem uses the exact same code and stencil as part one (see Figure 2). The metal plate in this problem is 10ft long (x) and 6ft wide (y). The bottom of the plate is held at 400°F, the top is held at 250°F, and the sides are both 150°F. The only difference between part four and five is that part four solves the PDE with a 1ft step-size and part five solves it with a 2in step-size.

$$Laplace's\ Equation\ (2D)$$

$$u_{xx} + u_{yy} = 0$$

$$u(0,y) = u(10,y) = 150°\,F$$

$$u(x,0) = 400°\,F\,,\ u(x,6) = 250°\,F$$

$$Solution:$$

$$u(x,y) = \sum\nolimits_{n=1}\left(B_n\left(100\left(\sin\left(\frac{n\pi x}{10}\right)\sinh\left(\frac{n\pi y}{10}\right)\right) + 250\left(\sin\left(\frac{n\pi x}{10}\right)\sinh\left(\frac{n\pi(6-y)}{10}\right)\right)\right)\right) + 150$$

$$B_n = \frac{2}{n\pi\sinh\left(\frac{6n\pi}{10}\right)}(1 - \cos(n\pi))$$

Figure 19: Metal Plate PDE for Parts 4 and 5

Figure 21 shows the numerical solution for part four, and Figure 20 shows the 200°F and 300°F isothermal curves. The locations of the isothermal curves make sense. The 300°F curve must be around the 400°F boundary because all of the other boundaries are less than 300°F, so the temperature will continue to decrease as it approaches the other boundaries. By similar logic, the 200°F curves must be closest to the 150°F boundaries since the other boundaries are greater than 150°F. Since there are two 150°F boundaries, there are two separate 200°F isothermal curves. These curves should be identical since the solution is symmetric over the plane, x = 5.

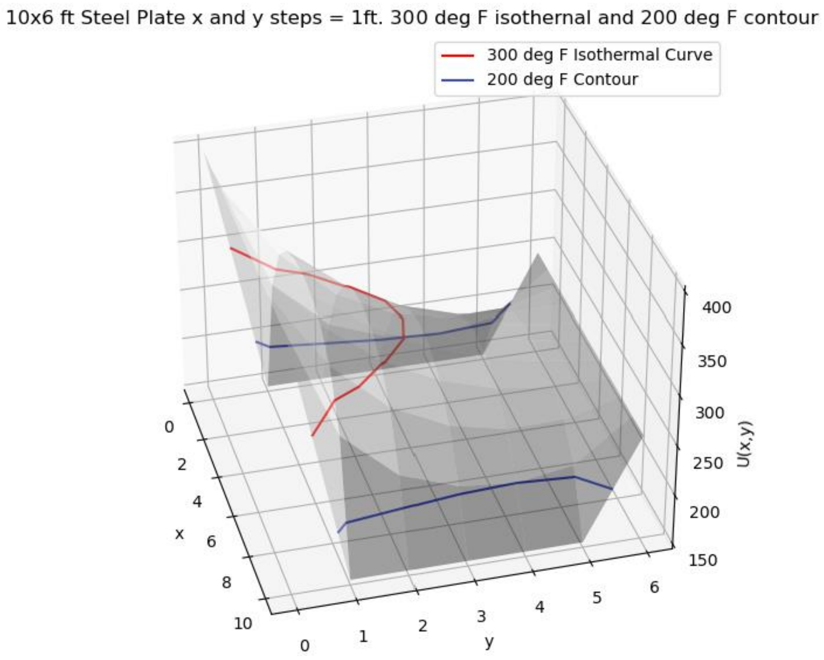


Figure 20: Contour Solution Graph for Part 4

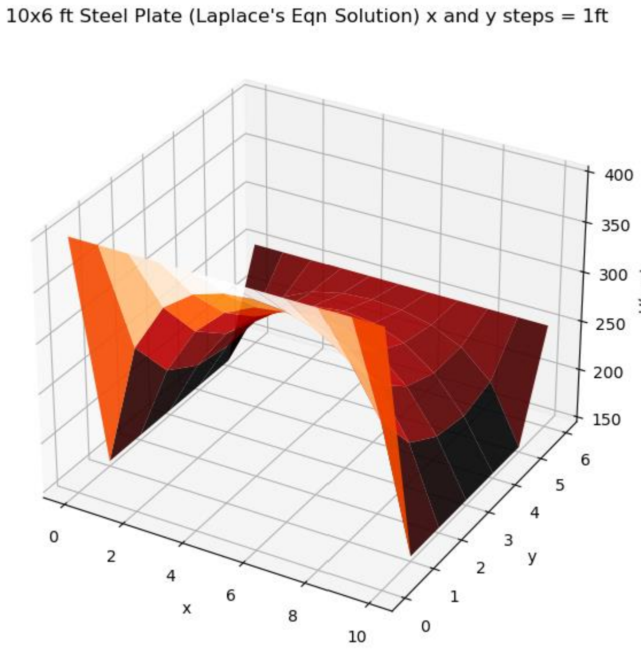


Figure 21: Solution Graph for Part 4

Figure 22 contains the solution graph for part five, while Figure 23 shows the 200°F and 300°F isothermal curves. Other than the size of the grid that makes up the solution, the graphs look the same as the previous ones.

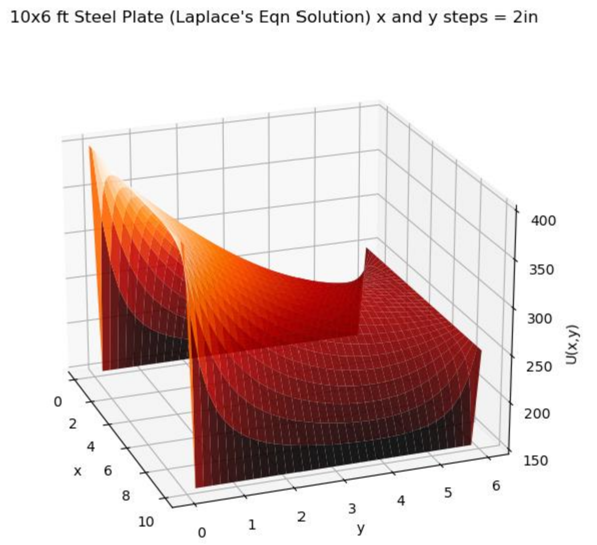


Figure 22: Solution Graph for Part 5

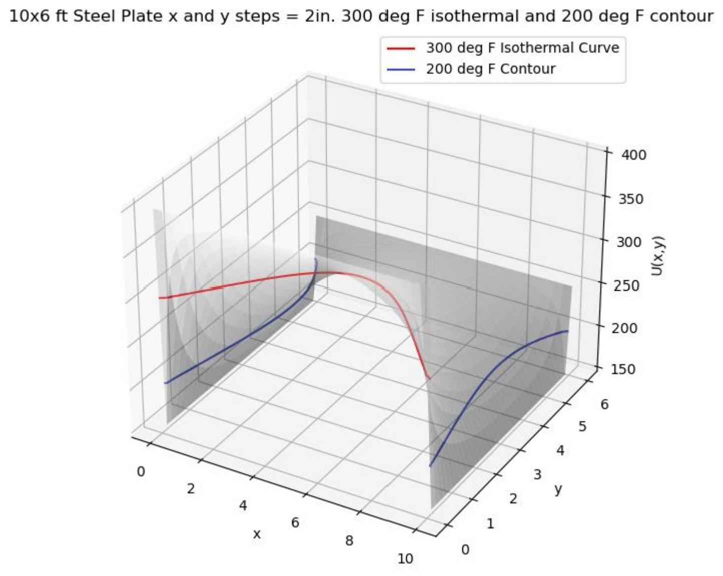


Figure 23: Contour Solution Graph for Part 5

Figure 24 is a graph of the exact solution for part four and five. Figure 25 shows the 200°F and 300°F isothermal curves. Comparing the exact solution graphs with the numerical solutions, they look identical. The solution was created with a sum of 60 values, though increasing the number of values did not seem to make it noticeably more accurate. The code to create and graph the exact solution is shown in Figure 36.

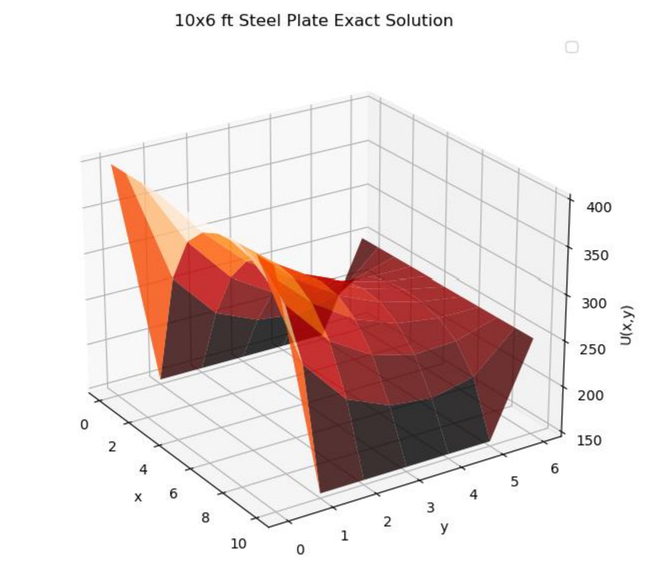


Figure 24: Exact Solution Graph for Part 4 and 5

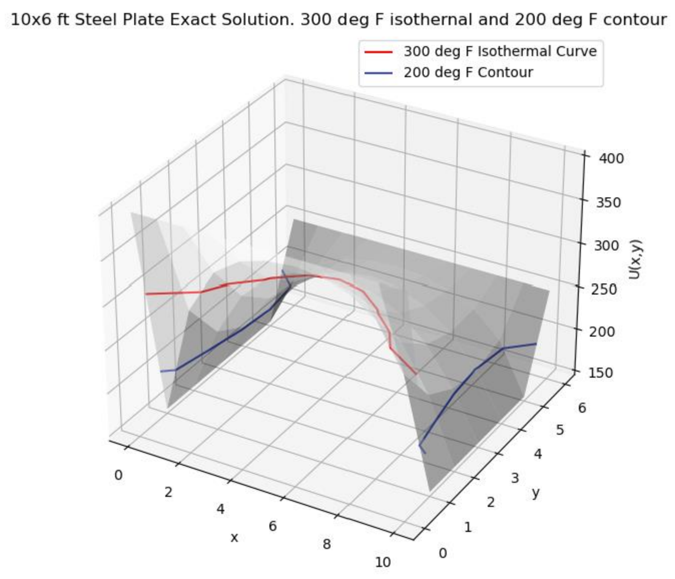


Figure 25: Exact Contour Solution Graph for Part 4 and 5

Figure 26 is an error graph comparing the exact solution and the numerical solution from part four. The error seems to be quite high, since the range of values is from -1.5 to 1.5. I double checked my code to make sure nothing was wrong, and both the A matrix and b vector seem to be correct, so I am not certain what is causing such large error. Even in Figure 28, which is a graph of the error for numerical solution with 100 x-steps and 60 y-steps, there are some large error-values. The maximum error even grows when compared to the part four solution. It is interesting that the errors tend to be concentrated around the corners of the solutions, since those are the only points that must account for two boundary conditions instead of one. Other than the corners the rest of the numerical solution converged to the exact solution. The code to create the numerical solution and error graphs is shown in Figure 37.

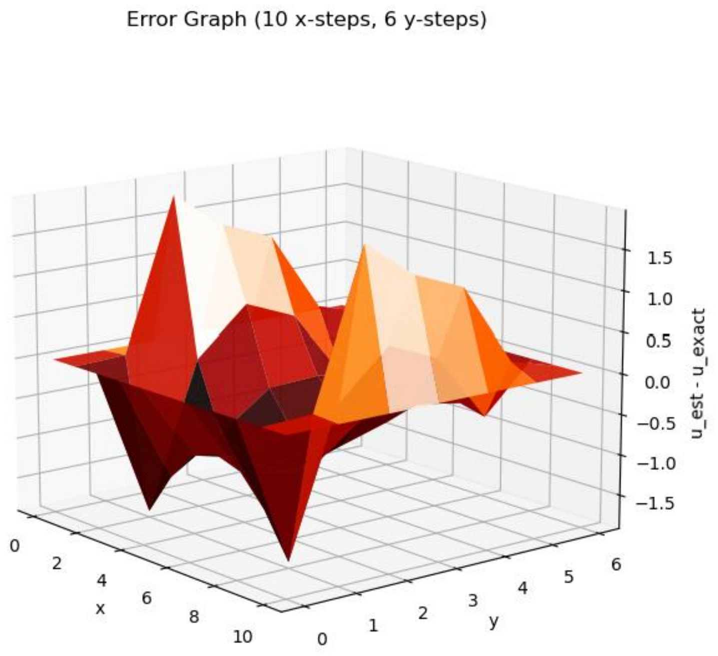


Figure 26: Error Graph for Part 4

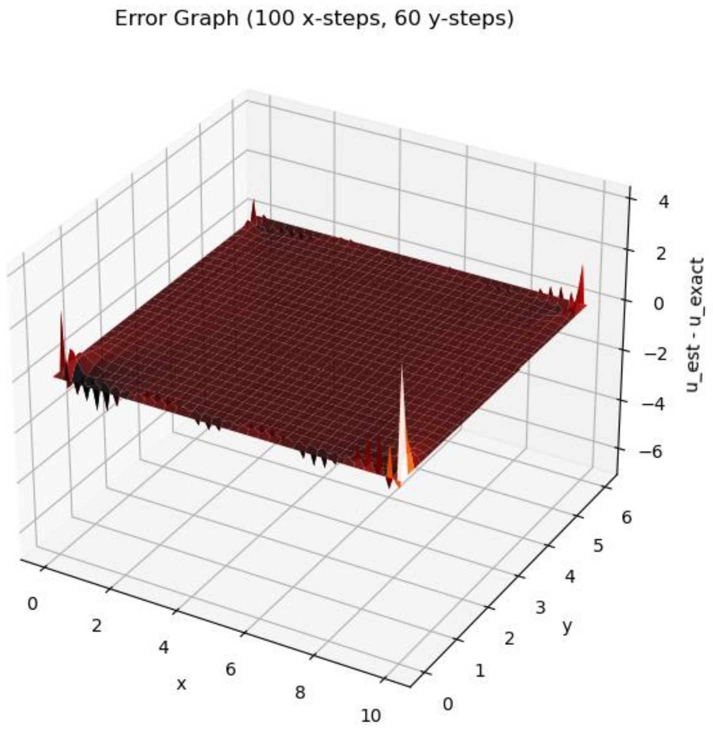


Figure 27: Extra Error Graph with 100 x-steps and 60 y-steps

Part 6

The objective of part six was to implement a finite difference method to solve Laplace's equation for a metal plate that has a square hole in it. In this case, the plate was 10ft long and 10ft wide, with a 4ft-by-4ft hole directly in the center. The addition of a hole changes the implementation of the numerical method because it must account for more boundary conditions when forming both A and b. Figure 28 shows the PDE and boundary conditions for this problem. The exact solution was not calculated.

$Laplace's\ Equation\ (2D)$

$u_{xx} + u_{yy} = 0$

$Exterior\ Rectangular\ Region:$

$u(0, y) = u(10, y) = 0°\ F$

$u(x, 0) = 0°\ F\ ,\ u(x, 10) = 100°\ F$

$Interior\ Rectangular\ Region:$

$u(3, y) = 0°\ F\ for\ 3 \leq y \leq 7$

$u(7, y) = 0°\ F\ for\ 3 \leq y \leq 7$

$u(x, 3) = 0°\ F\ for\ 3 \leq x \leq 7$

$u(x, 7) = 100°\ F\ for\ 3 \leq y \leq 7$

Figure 28: Metal Plate with Square Hole PDE for Part 6

Figure 29 shows the numerical solution graph for this PDE with 50 steps in x and y. Given the boundary conditions, the solution is what you would expect. The top inner and outer boundaries are both held at 100°F, and the rest of the boundaries are at 0°F. Below the 100°F interior boundary, the solution quickly drops to zero. Between the two 100°F boundaries, the solution dips slightly below 100°F, with the lowest point being in the middle of the two boundaries. The method code and the code to create this graph can be found in Figure 38.

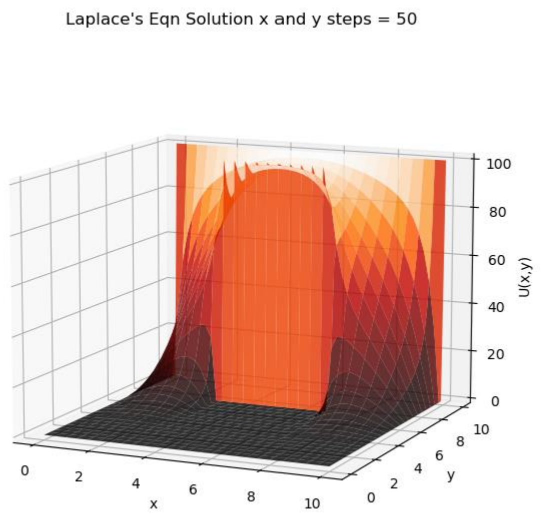
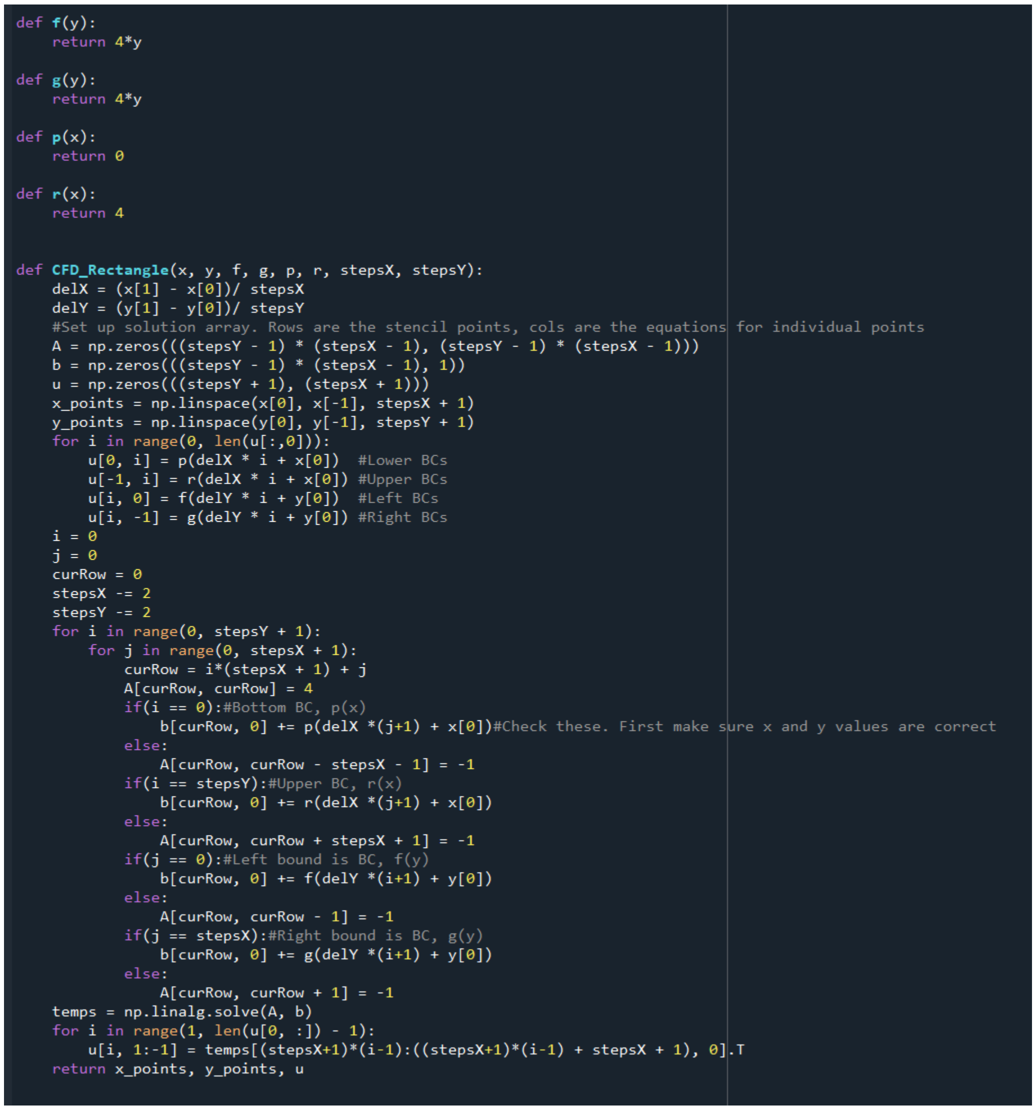


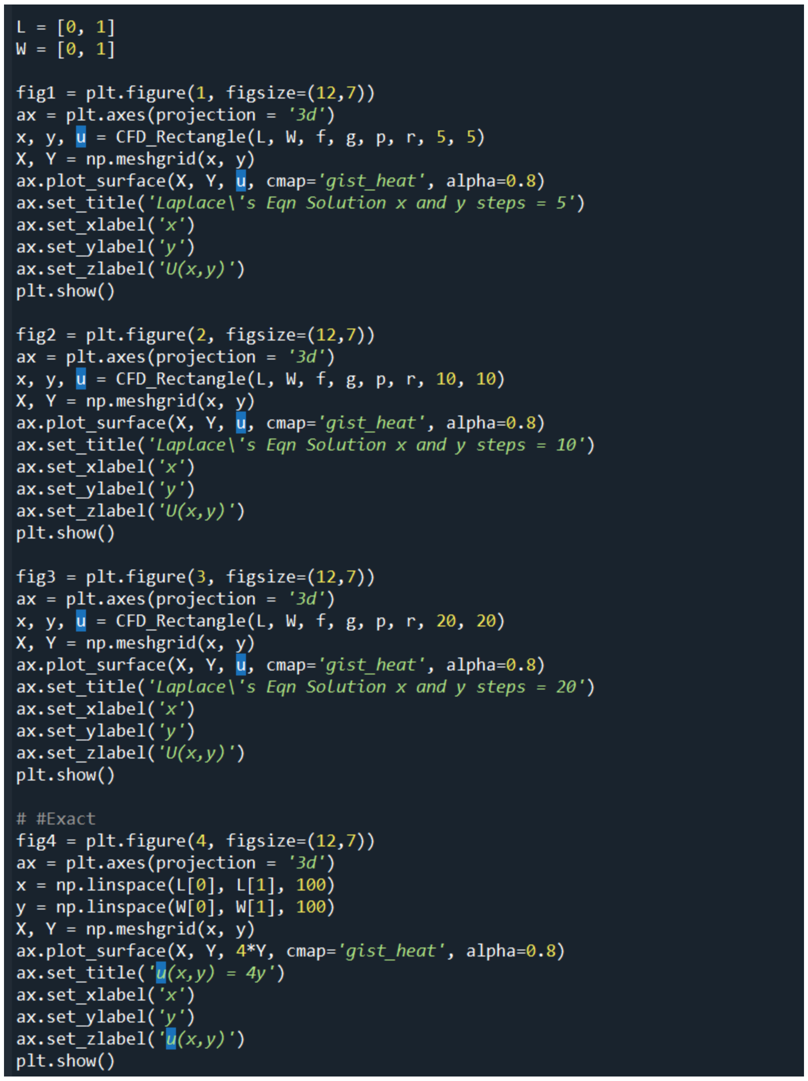
Figure 29: Part 6 Numerical Solution Graph

Appendix



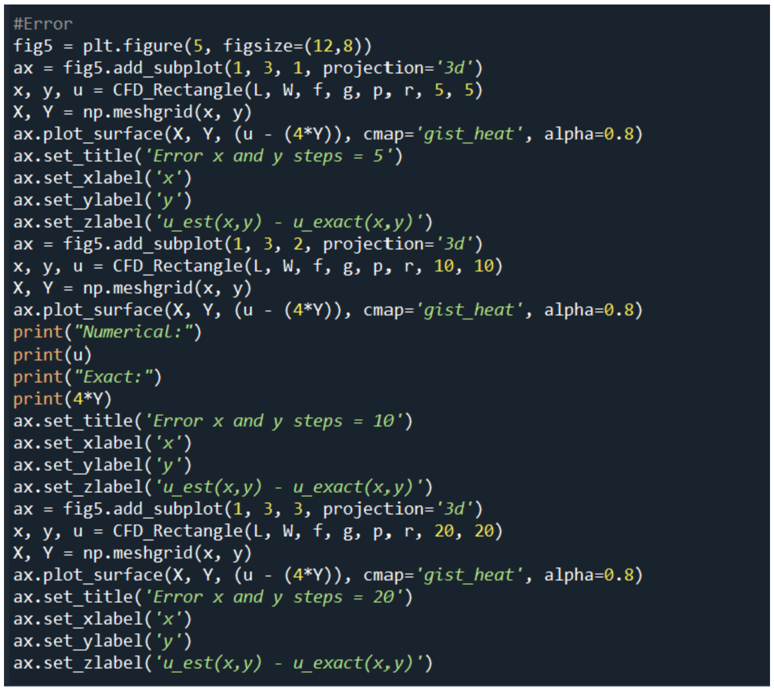
```python
def f(y):
    return 4*y

def g(y):
    return 4*y

def p(x):
    return 0

def r(x):
    return 4


def CFD_Rectangle(x, y, f, g, p, r, stepsX, stepsY):
    delX = (x[1] - x[0])/ stepsX
    delY = (y[1] - y[0])/ stepsY
    #Set up solution array. Rows are the stencil points, cols are the equations for individual points
    A = np.zeros(((stepsY - 1) * (stepsX - 1), (stepsY - 1) * (stepsX - 1)))
    b = np.zeros(((stepsY - 1) * (stepsX - 1), 1))
    u = np.zeros(((stepsY + 1), (stepsX + 1)))
    x_points = np.linspace(x[0], x[-1], stepsX + 1)
    y_points = np.linspace(y[0], y[-1], stepsY + 1)
    for i in range(0, len(u[:,0])):
        u[0, i] = p(delX * i + x[0])  #Lower BCs
        u[-1, i] = r(delX * i + x[0]) #Upper BCs
        u[i, 0] = f(delY * i + y[0])  #Left BCs
        u[i, -1] = g(delY * i + y[0]) #Right BCs
    i = 0
    j = 0
    curRow = 0
    stepsX -= 2
    stepsY -= 2
    for i in range(0, stepsY + 1):
        for j in range(0, stepsX + 1):
            curRow = i*(stepsX + 1) + j
            A[curRow, curRow] = 4
            if(i == 0):#Bottom BC, p(x)
                b[curRow, 0] += p(delX *(j+1) + x[0])#Check these. First make sure x and y values are correct
            else:
                A[curRow, curRow - stepsX - 1] = -1
            if(i == stepsY):#Upper BC, r(x)
                b[curRow, 0] += r(delX *(j+1) + x[0])
            else:
                A[curRow, curRow + stepsX + 1] = -1
            if(j == 0):#Left bound is BC, f(y)
                b[curRow, 0] += f(delY *(i+1) + y[0])
            else:
                A[curRow, curRow - 1] = -1
            if(j == stepsX):#Right bound is BC, g(y)
                b[curRow, 0] += g(delY *(i+1) + y[0])
            else:
                A[curRow, curRow + 1] = -1
    temps = np.linalg.solve(A, b)
    for i in range(1, len(u[0, :]) - 1):
        u[i, 1:-1] = temps[(stepsX+1)*(i-1):((stepsX+1)*(i-1) + stepsX + 1), 0].T
    return x_points, y_points, u
```

Figure 30: Centered Finite Difference Method Code for 2D Laplace's Equation



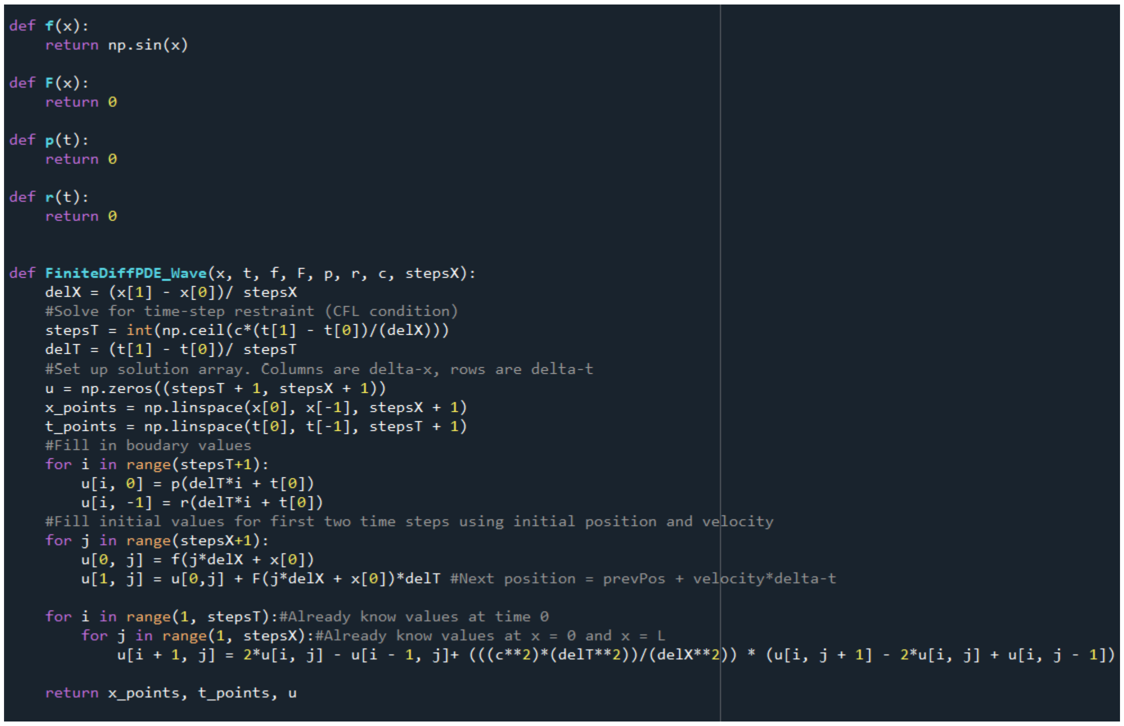
```python
L = [0, 1]
W = [0, 1]

fig1 = plt.figure(1, figsize=(12,7))
ax = plt.axes(projection = '3d')
x, y, u = CFD_Rectangle(L, W, f, g, p, r, 5, 5)
X, Y = np.meshgrid(x, y)
ax.plot_surface(X, Y, u, cmap='gist_heat', alpha=0.8)
ax.set_title('Laplace\'s Eqn Solution x and y steps = 5')
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('U(x,y)')
plt.show()

fig2 = plt.figure(2, figsize=(12,7))
ax = plt.axes(projection = '3d')
x, y, u = CFD_Rectangle(L, W, f, g, p, r, 10, 10)
X, Y = np.meshgrid(x, y)
ax.plot_surface(X, Y, u, cmap='gist_heat', alpha=0.8)
ax.set_title('Laplace\'s Eqn Solution x and y steps = 10')
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('U(x,y)')
plt.show()

fig3 = plt.figure(3, figsize=(12,7))
ax = plt.axes(projection = '3d')
x, y, u = CFD_Rectangle(L, W, f, g, p, r, 20, 20)
X, Y = np.meshgrid(x, y)
ax.plot_surface(X, Y, u, cmap='gist_heat', alpha=0.8)
ax.set_title('Laplace\'s Eqn Solution x and y steps = 20')
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('U(x,y)')
plt.show()

# #Exact
fig4 = plt.figure(4, figsize=(12,7))
ax = plt.axes(projection = '3d')
x = np.linspace(L[0], L[1], 100)
y = np.linspace(W[0], W[1], 100)
X, Y = np.meshgrid(x, y)
ax.plot_surface(X, Y, 4*Y, cmap='gist_heat', alpha=0.8)
ax.set_title('u(x,y) = 4y')
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('u(x,y)')
plt.show()
```

Figure 31: Graphing Code for Part 1



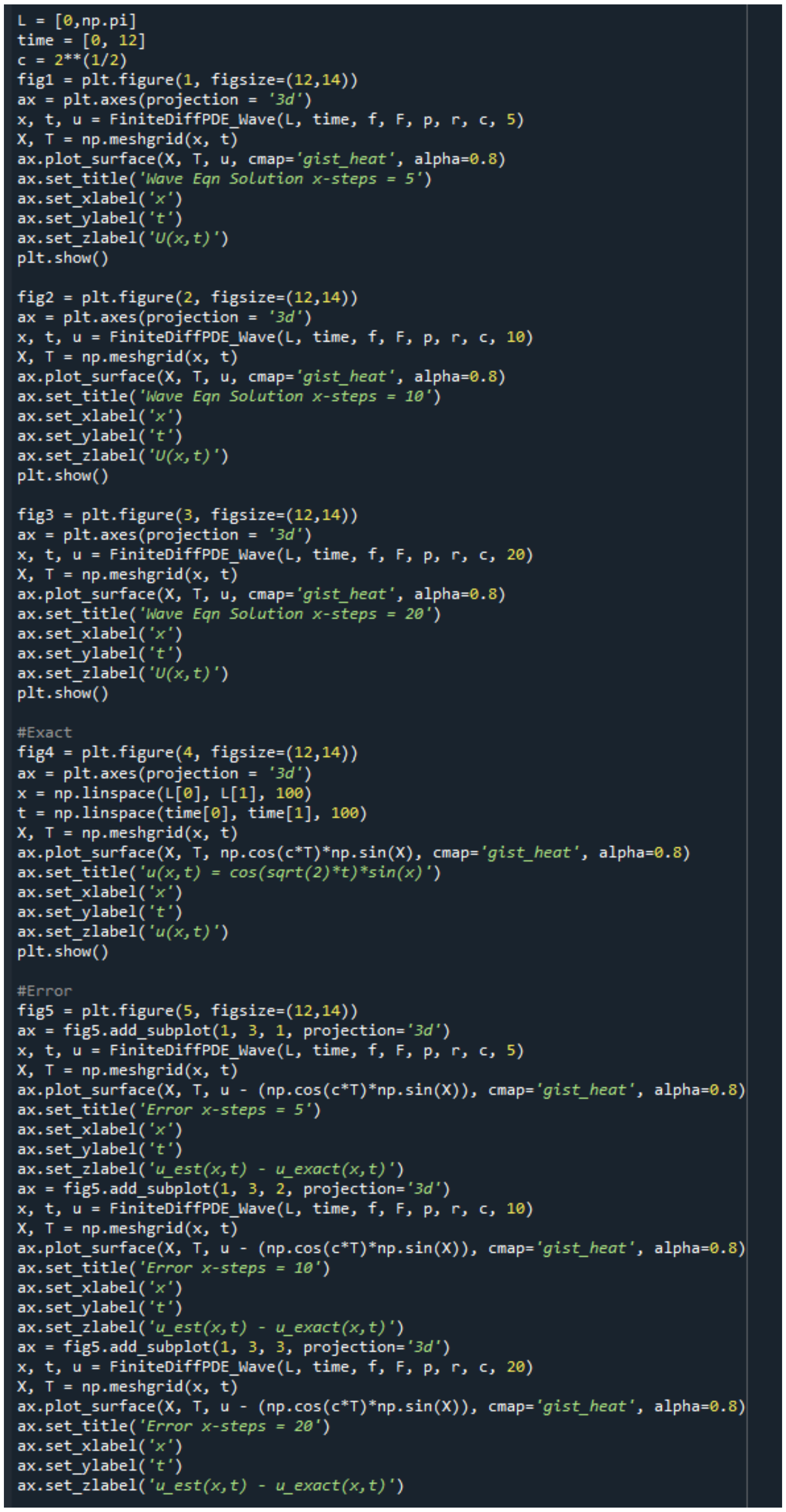
```python
#Error
fig5 = plt.figure(5, figsize=(12,8))
ax = fig5.add_subplot(1, 3, 1, projection='3d')
x, y, u = CFD_Rectangle(L, W, f, g, p, r, 5, 5)
X, Y = np.meshgrid(x, y)
ax.plot_surface(X, Y, (u - (4*Y)), cmap='gist_heat', alpha=0.8)
ax.set_title('Error x and y steps = 5')
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('u_est(x,y) - u_exact(x,y)')
ax = fig5.add_subplot(1, 3, 2, projection='3d')
x, y, u = CFD_Rectangle(L, W, f, g, p, r, 10, 10)
X, Y = np.meshgrid(x, y)
ax.plot_surface(X, Y, (u - (4*Y)), cmap='gist_heat', alpha=0.8)
print("Numerical:")
print(u)
print("Exact:")
print(4*Y)
ax.set_title('Error x and y steps = 10')
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('u_est(x,y) - u_exact(x,y)')
ax = fig5.add_subplot(1, 3, 3, projection='3d')
x, y, u = CFD_Rectangle(L, W, f, g, p, r, 20, 20)
X, Y = np.meshgrid(x, y)
ax.plot_surface(X, Y, (u - (4*Y)), cmap='gist_heat', alpha=0.8)
ax.set_title('Error x and y steps = 20')
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('u_est(x,y) - u_exact(x,y)')
```

Figure 32: Error Graphing Code for Part 1



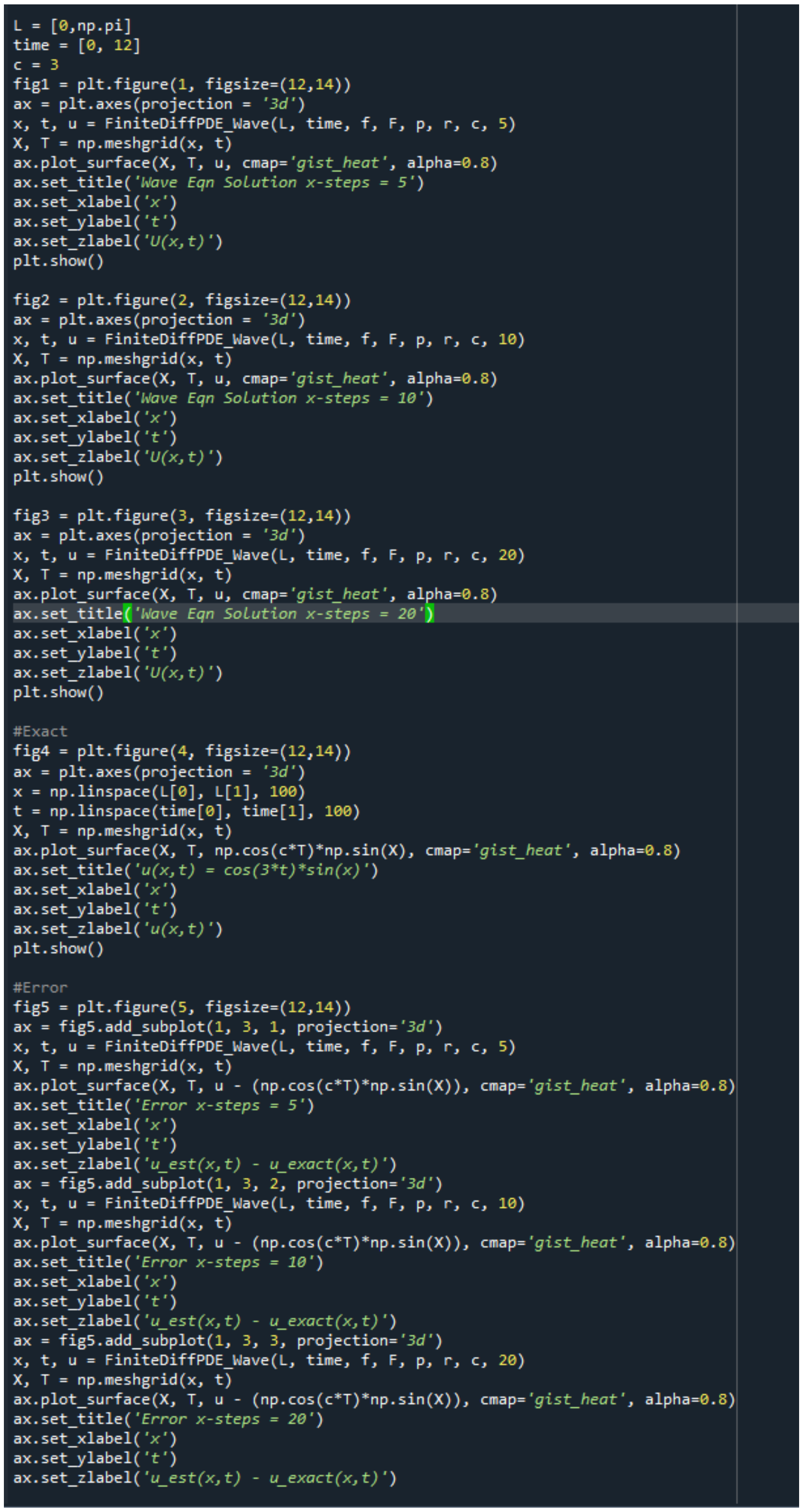
```python
def f(x):
    return np.sin(x)

def F(x):
    return 0

def p(t):
    return 0

def r(t):
    return 0


def FiniteDiffPDE_Wave(x, t, f, F, p, r, c, stepsX):
    delX = (x[1] - x[0])/ stepsX
    #Solve for time-step restraint (CFL condition)
    stepsT = int(np.ceil(c*(t[1] - t[0])/(delX)))
    delT = (t[1] - t[0])/ stepsT
    #Set up solution array. Columns are delta-x, rows are delta-t
    u = np.zeros((stepsT + 1, stepsX + 1))
    x_points = np.linspace(x[0], x[-1], stepsX + 1)
    t_points = np.linspace(t[0], t[-1], stepsT + 1)
    #Fill in boudary values
    for i in range(stepsT+1):
        u[i, 0] = p(delT*i + t[0])
        u[i, -1] = r(delT*i + t[0])
    #Fill initial values for first two time steps using initial position and velocity
    for j in range(stepsX+1):
        u[0, j] = f(j*delX + x[0])
        u[1, j] = u[0,j] + F(j*delX + x[0])*delT #Next position = prevPos + velocity*delta-t

    for i in range(1, stepsT):#Already know values at time 0
        for j in range(1, stepsX):#Already know values at x = 0 and x = L
            u[i + 1, j] = 2*u[i, j] - u[i - 1, j]+ (((c**2)*(delT**2))/(delX**2)) * (u[i, j + 1] - 2*u[i, j] + u[i, j - 1])

    return x_points, t_points, u
```

Figure 33: Centered Finite Difference Method Code for 1D Wave



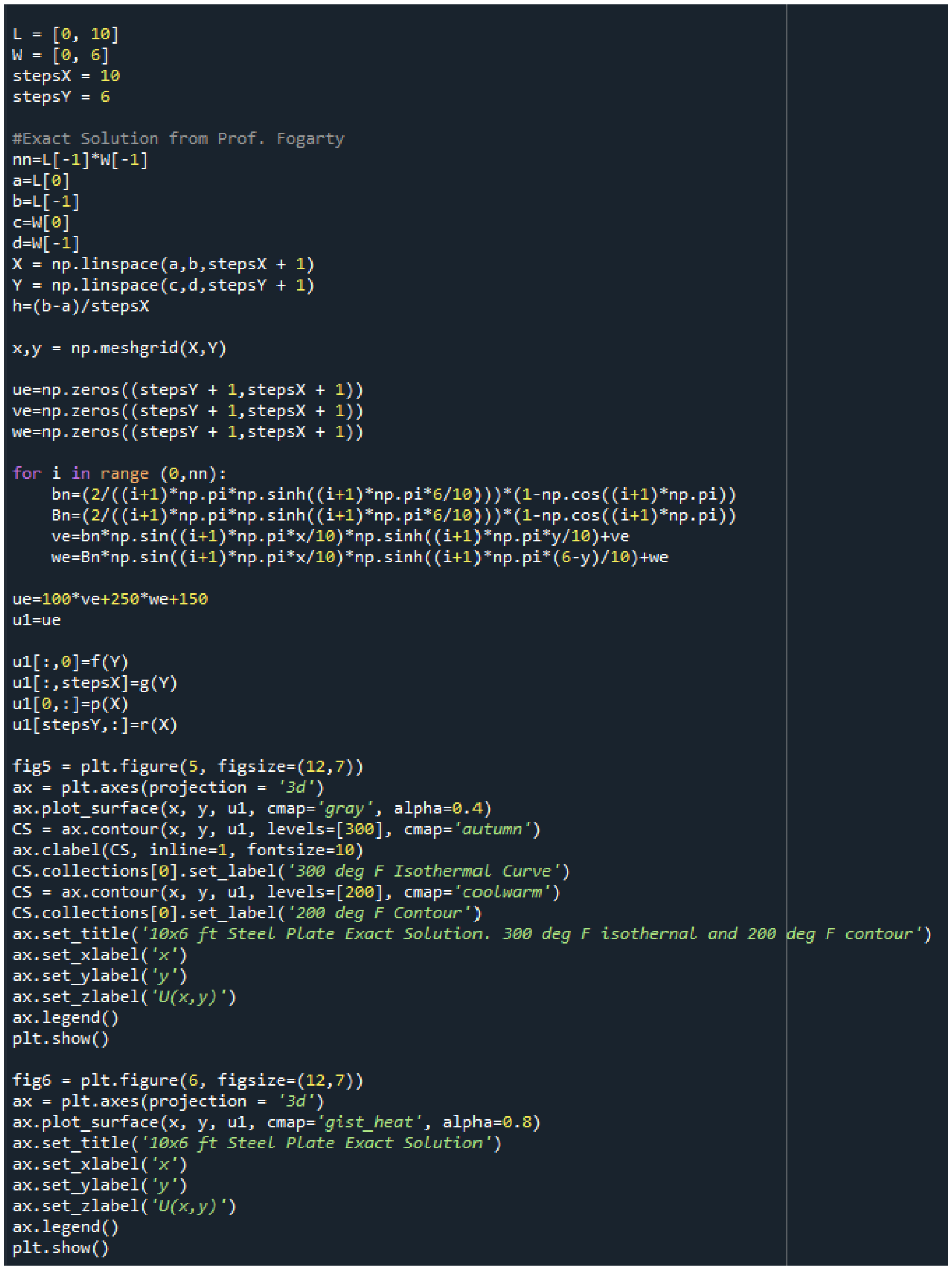
```
L = [0,np.pi]
time = [0, 12]
c = 2**(1/2)
fig1 = plt.figure(1, figsize=(12,14))
ax = plt.axes(projection = '3d')
x, t, u = FiniteDiffPDE_Wave(L, time, f, F, p, r, c, 5)
X, T = np.meshgrid(x, t)
ax.plot_surface(X, T, u, cmap='gist_heat', alpha=0.8)
ax.set_title('Wave Eqn Solution x-steps = 5')
ax.set_xlabel('x')
ax.set_ylabel('t')
ax.set_zlabel('U(x,t)')
plt.show()

fig2 = plt.figure(2, figsize=(12,14))
ax = plt.axes(projection = '3d')
x, t, u = FiniteDiffPDE_Wave(L, time, f, F, p, r, c, 10)
X, T = np.meshgrid(x, t)
ax.plot_surface(X, T, u, cmap='gist_heat', alpha=0.8)
ax.set_title('Wave Eqn Solution x-steps = 10')
ax.set_xlabel('x')
ax.set_ylabel('t')
ax.set_zlabel('U(x,t)')
plt.show()

fig3 = plt.figure(3, figsize=(12,14))
ax = plt.axes(projection = '3d')
x, t, u = FiniteDiffPDE_Wave(L, time, f, F, p, r, c, 20)
X, T = np.meshgrid(x, t)
ax.plot_surface(X, T, u, cmap='gist_heat', alpha=0.8)
ax.set_title('Wave Eqn Solution x-steps = 20')
ax.set_xlabel('x')
ax.set_ylabel('t')
ax.set_zlabel('U(x,t)')
plt.show()

#Exact
fig4 = plt.figure(4, figsize=(12,14))
ax = plt.axes(projection = '3d')
x = np.linspace(L[0], L[1], 100)
t = np.linspace(time[0], time[1], 100)
X, T = np.meshgrid(x, t)
ax.plot_surface(X, T, np.cos(c*T)*np.sin(X), cmap='gist_heat', alpha=0.8)
ax.set_title('u(x,t) = cos(sqrt(2)*t)*sin(x)')
ax.set_xlabel('x')
ax.set_ylabel('t')
ax.set_zlabel('u(x,t)')
plt.show()

#Error
fig5 = plt.figure(5, figsize=(12,14))
ax = fig5.add_subplot(1, 3, 1, projection='3d')
x, t, u = FiniteDiffPDE_Wave(L, time, f, F, p, r, c, 5)
X, T = np.meshgrid(x, t)
ax.plot_surface(X, T, u - (np.cos(c*T)*np.sin(X)), cmap='gist_heat', alpha=0.8)
ax.set_title('Error x-steps = 5')
ax.set_xlabel('x')
ax.set_ylabel('t')
ax.set_zlabel('u_est(x,t) - u_exact(x,t)')
ax = fig5.add_subplot(1, 3, 2, projection='3d')
x, t, u = FiniteDiffPDE_Wave(L, time, f, F, p, r, c, 10)
X, T = np.meshgrid(x, t)
ax.plot_surface(X, T, u - (np.cos(c*T)*np.sin(X)), cmap='gist_heat', alpha=0.8)
ax.set_title('Error x-steps = 10')
ax.set_xlabel('x')
ax.set_ylabel('t')
ax.set_zlabel('u_est(x,t) - u_exact(x,t)')
ax = fig5.add_subplot(1, 3, 3, projection='3d')
x, t, u = FiniteDiffPDE_Wave(L, time, f, F, p, r, c, 20)
X, T = np.meshgrid(x, t)
ax.plot_surface(X, T, u - (np.cos(c*T)*np.sin(X)), cmap='gist_heat', alpha=0.8)
ax.set_title('Error x-steps = 20')
ax.set_xlabel('x')
ax.set_ylabel('t')
ax.set_zlabel('u_est(x,t) - u_exact(x,t)')
```

Figure 34: Graphing Code for Part 2



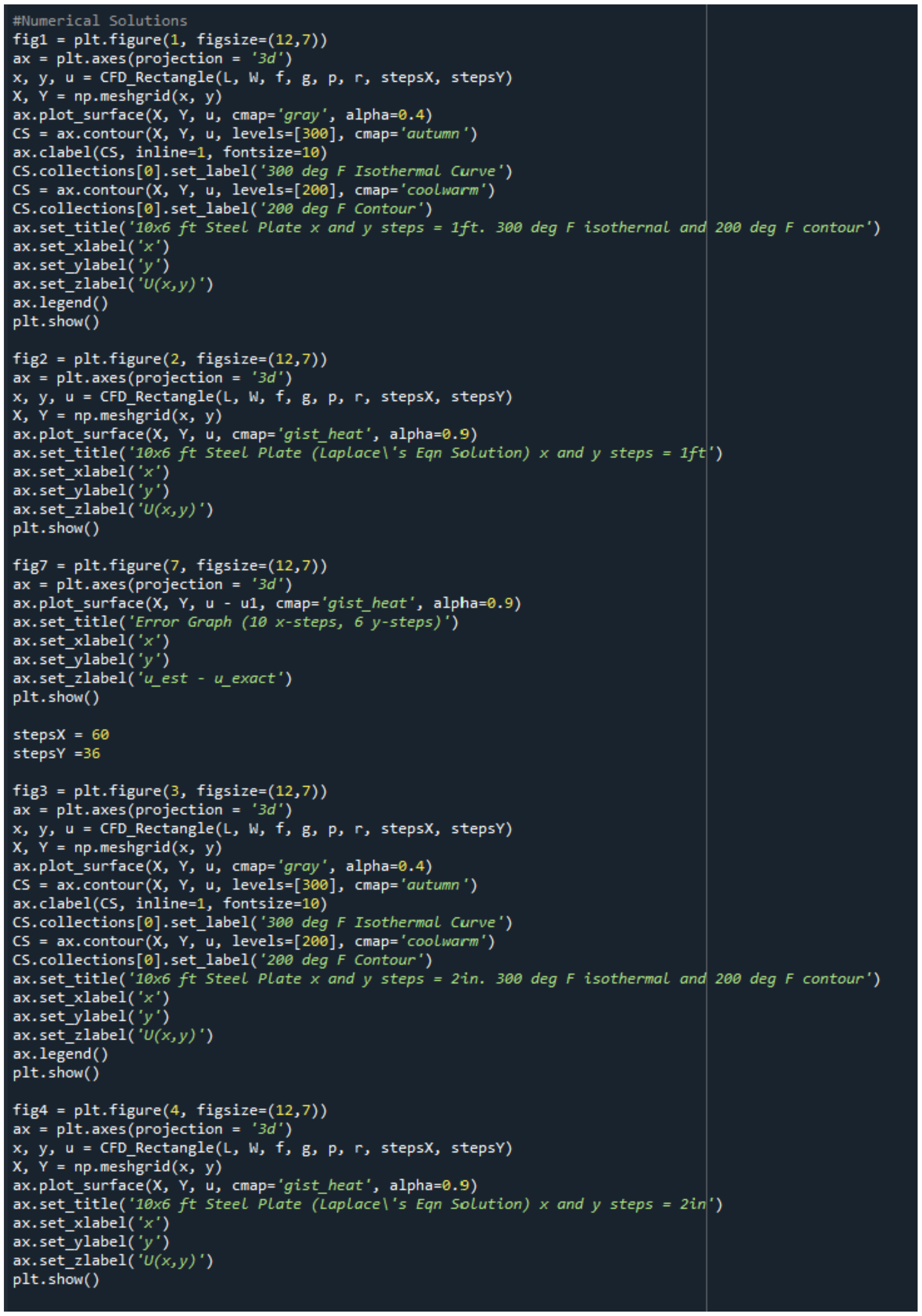
```python
L = [0,np.pi]
time = [0, 12]
c = 3
fig1 = plt.figure(1, figsize=(12,14))
ax = plt.axes(projection = '3d')
x, t, u = FiniteDiffPDE_Wave(L, time, f, F, p, r, c, 5)
X, T = np.meshgrid(x, t)
ax.plot_surface(X, T, u, cmap='gist_heat', alpha=0.8)
ax.set_title('Wave Eqn Solution x-steps = 5')
ax.set_xlabel('x')
ax.set_ylabel('t')
ax.set_zlabel('U(x,t)')
plt.show()

fig2 = plt.figure(2, figsize=(12,14))
ax = plt.axes(projection = '3d')
x, t, u = FiniteDiffPDE_Wave(L, time, f, F, p, r, c, 10)
X, T = np.meshgrid(x, t)
ax.plot_surface(X, T, u, cmap='gist_heat', alpha=0.8)
ax.set_title('Wave Eqn Solution x-steps = 10')
ax.set_xlabel('x')
ax.set_ylabel('t')
ax.set_zlabel('U(x,t)')
plt.show()

fig3 = plt.figure(3, figsize=(12,14))
ax = plt.axes(projection = '3d')
x, t, u = FiniteDiffPDE_Wave(L, time, f, F, p, r, c, 20)
X, T = np.meshgrid(x, t)
ax.plot_surface(X, T, u, cmap='gist_heat', alpha=0.8)
ax.set_title('Wave Eqn Solution x-steps = 20')
ax.set_xlabel('x')
ax.set_ylabel('t')
ax.set_zlabel('U(x,t)')
plt.show()

#Exact
fig4 = plt.figure(4, figsize=(12,14))
ax = plt.axes(projection = '3d')
x = np.linspace(L[0], L[1], 100)
t = np.linspace(time[0], time[1], 100)
X, T = np.meshgrid(x, t)
ax.plot_surface(X, T, np.cos(c*T)*np.sin(X), cmap='gist_heat', alpha=0.8)
ax.set_title('u(x,t) = cos(3*t)*sin(x)')
ax.set_xlabel('x')
ax.set_ylabel('t')
ax.set_zlabel('u(x,t)')
plt.show()

#Error
fig5 = plt.figure(5, figsize=(12,14))
ax = fig5.add_subplot(1, 3, 1, projection='3d')
x, t, u = FiniteDiffPDE_Wave(L, time, f, F, p, r, c, 5)
X, T = np.meshgrid(x, t)
ax.plot_surface(X, T, u - (np.cos(c*T)*np.sin(X)), cmap='gist_heat', alpha=0.8)
ax.set_title('Error x-steps = 5')
ax.set_xlabel('x')
ax.set_ylabel('t')
ax.set_zlabel('u_est(x,t) - u_exact(x,t)')
ax = fig5.add_subplot(1, 3, 2, projection='3d')
x, t, u = FiniteDiffPDE_Wave(L, time, f, F, p, r, c, 10)
X, T = np.meshgrid(x, t)
ax.plot_surface(X, T, u - (np.cos(c*T)*np.sin(X)), cmap='gist_heat', alpha=0.8)
ax.set_title('Error x-steps = 10')
ax.set_xlabel('x')
ax.set_ylabel('t')
ax.set_zlabel('u_est(x,t) - u_exact(x,t)')
ax = fig5.add_subplot(1, 3, 3, projection='3d')
x, t, u = FiniteDiffPDE_Wave(L, time, f, F, p, r, c, 20)
X, T = np.meshgrid(x, t)
ax.plot_surface(X, T, u - (np.cos(c*T)*np.sin(X)), cmap='gist_heat', alpha=0.8)
ax.set_title('Error x-steps = 20')
ax.set_xlabel('x')
ax.set_ylabel('t')
ax.set_zlabel('u_est(x,t) - u_exact(x,t)')
```

Figure 35: Graphing Code for Part 3



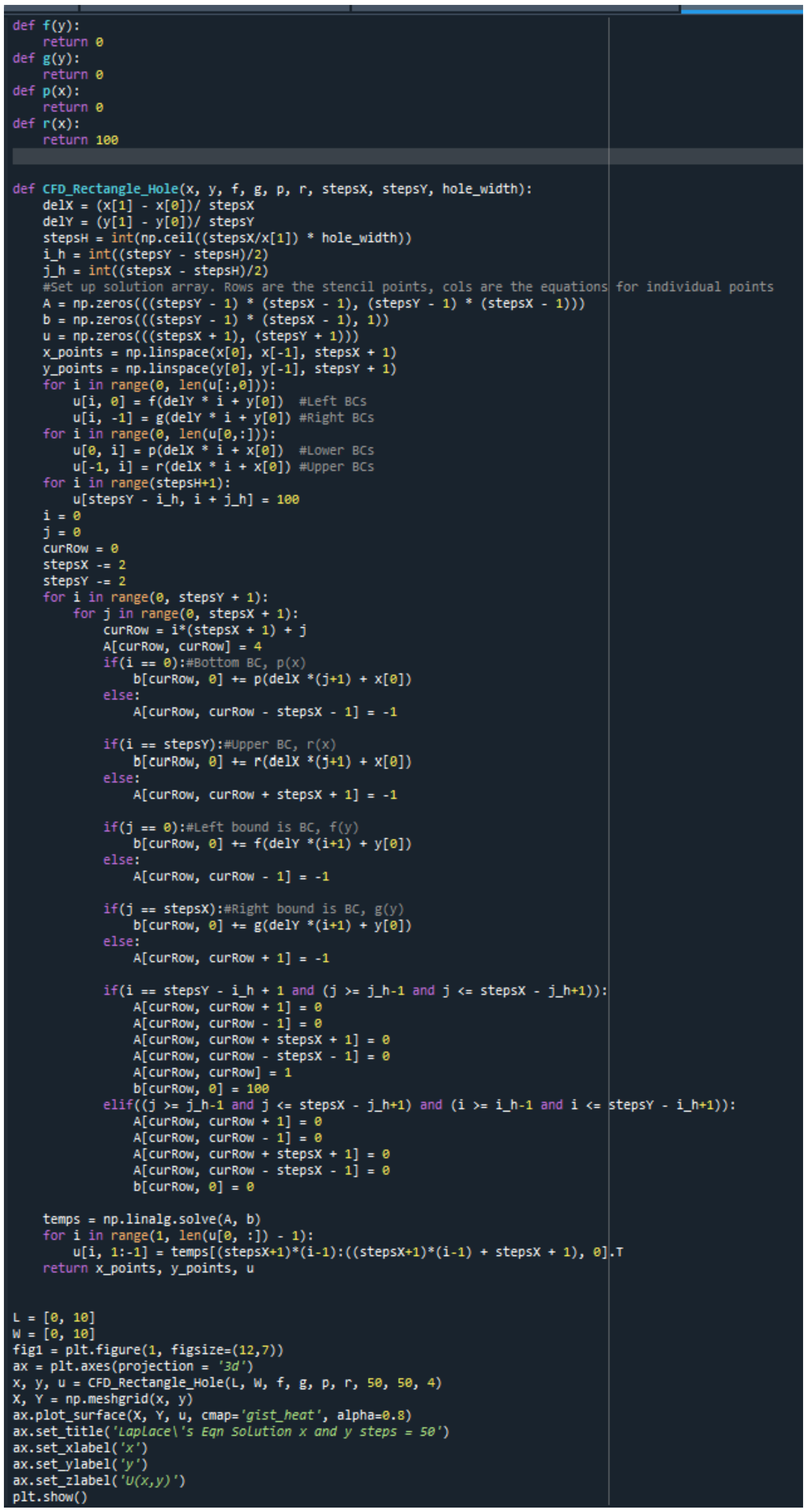
```
L = [0, 10]
W = [0, 6]
stepsX = 10
stepsY = 6

#Exact Solution from Prof. Fogarty
nn=L[-1]*W[-1]
a=L[0]
b=L[-1]
c=W[0]
d=W[-1]
X = np.linspace(a,b,stepsX + 1)
Y = np.linspace(c,d,stepsY + 1)
h=(b-a)/stepsX

x,y = np.meshgrid(X,Y)

ue=np.zeros((stepsY + 1,stepsX + 1))
ve=np.zeros((stepsY + 1,stepsX + 1))
we=np.zeros((stepsY + 1,stepsX + 1))

for i in range (0,nn):
    bn=(2/((i+1)*np.pi*np.sinh((i+1)*np.pi*6/10)))*(1-np.cos((i+1)*np.pi))
    Bn=(2/((i+1)*np.pi*np.sinh((i+1)*np.pi*6/10)))*(1-np.cos((i+1)*np.pi))
    ve=bn*np.sin((i+1)*np.pi*x/10)*np.sinh((i+1)*np.pi*y/10)+ve
    we=Bn*np.sin((i+1)*np.pi*x/10)*np.sinh((i+1)*np.pi*(6-y)/10)+we

ue=100*ve+250*we+150
u1=ue

u1[:,0]=f(Y)
u1[:,stepsX]=g(Y)
u1[0,:]=p(X)
u1[stepsY,:]=r(X)

fig5 = plt.figure(5, figsize=(12,7))
ax = plt.axes(projection = '3d')
ax.plot_surface(x, y, u1, cmap='gray', alpha=0.4)
CS = ax.contour(x, y, u1, levels=[300], cmap='autumn')
ax.clabel(CS, inline=1, fontsize=10)
CS.collections[0].set_label('300 deg F Isothermal Curve')
CS = ax.contour(x, y, u1, levels=[200], cmap='coolwarm')
CS.collections[0].set_label('200 deg F Contour')
ax.set_title('10x6 ft Steel Plate Exact Solution. 300 deg F isothernal and 200 deg F contour')
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('U(x,y)')
ax.legend()
plt.show()

fig6 = plt.figure(6, figsize=(12,7))
ax = plt.axes(projection = '3d')
ax.plot_surface(x, y, u1, cmap='gist_heat', alpha=0.8)
ax.set_title('10x6 ft Steel Plate Exact Solution')
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('U(x,y)')
ax.legend()
plt.show()
```

Figure 36: Exact Solution Graphing Code for Part 4



```
#Numerical Solutions
fig1 = plt.figure(1, figsize=(12,7))
ax = plt.axes(projection = '3d')
x, y, u = CFD_Rectangle(L, W, f, g, p, r, stepsX, stepsY)
X, Y = np.meshgrid(x, y)
ax.plot_surface(X, Y, u, cmap='gray', alpha=0.4)
CS = ax.contour(X, Y, u, levels=[300], cmap='autumn')
ax.clabel(CS, inline=1, fontsize=10)
CS.collections[0].set_label('300 deg F Isothermal Curve')
CS = ax.contour(X, Y, u, levels=[200], cmap='coolwarm')
CS.collections[0].set_label('200 deg F Contour')
ax.set_title('10x6 ft Steel Plate x and y steps = 1ft. 300 deg F isothernal and 200 deg F contour')
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('U(x,y)')
ax.legend()
plt.show()

fig2 = plt.figure(2, figsize=(12,7))
ax = plt.axes(projection = '3d')
x, y, u = CFD_Rectangle(L, W, f, g, p, r, stepsX, stepsY)
X, Y = np.meshgrid(x, y)
ax.plot_surface(X, Y, u, cmap='gist_heat', alpha=0.9)
ax.set_title('10x6 ft Steel Plate (Laplace\'s Eqn Solution) x and y steps = 1ft')
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('U(x,y)')
plt.show()

fig7 = plt.figure(7, figsize=(12,7))
ax = plt.axes(projection = '3d')
ax.plot_surface(X, Y, u - u1, cmap='gist_heat', alpha=0.9)
ax.set_title('Error Graph (10 x-steps, 6 y-steps)')
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('u_est - u_exact')
plt.show()

stepsX = 60
stepsY =36

fig3 = plt.figure(3, figsize=(12,7))
ax = plt.axes(projection = '3d')
x, y, u = CFD_Rectangle(L, W, f, g, p, r, stepsX, stepsY)
X, Y = np.meshgrid(x, y)
ax.plot_surface(X, Y, u, cmap='gray', alpha=0.4)
CS = ax.contour(X, Y, u, levels=[300], cmap='autumn')
ax.clabel(CS, inline=1, fontsize=10)
CS.collections[0].set_label('300 deg F Isothermal Curve')
CS = ax.contour(X, Y, u, levels=[200], cmap='coolwarm')
CS.collections[0].set_label('200 deg F Contour')
ax.set_title('10x6 ft Steel Plate x and y steps = 2in. 300 deg F isothernal and 200 deg F contour')
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('U(x,y)')
ax.legend()
plt.show()

fig4 = plt.figure(4, figsize=(12,7))
ax = plt.axes(projection = '3d')
x, y, u = CFD_Rectangle(L, W, f, g, p, r, stepsX, stepsY)
X, Y = np.meshgrid(x, y)
ax.plot_surface(X, Y, u, cmap='gist_heat', alpha=0.9)
ax.set_title('10x6 ft Steel Plate (Laplace\'s Eqn Solution) x and y steps = 2in')
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('U(x,y)')
plt.show()
```

Figure 37: Numerical Solution Graphing Code for Parts 4 and 5



```python
def f(y):
    return 0
def g(y):
    return 0
def p(x):
    return 0
def r(x):
    return 100


def CFD_Rectangle_Hole(x, y, f, g, p, r, stepsX, stepsY, hole_width):
    delX = (x[1] - x[0])/ stepsX
    delY = (y[1] - y[0])/ stepsY
    stepsH = int(np.ceil((stepsX/x[1]) * hole_width))
    i_h = int((stepsY - stepsH)/2)
    j_h = int((stepsX - stepsH)/2)
    #Set up solution array. Rows are the stencil points, cols are the equations for individual points
    A = np.zeros(((stepsY - 1) * (stepsX - 1), (stepsY - 1) * (stepsX - 1)))
    b = np.zeros(((stepsY - 1) * (stepsX - 1), 1))
    u = np.zeros(((stepsX + 1), (stepsY + 1)))
    x_points = np.linspace(x[0], x[-1], stepsX + 1)
    y_points = np.linspace(y[0], y[-1], stepsY + 1)
    for i in range(0, len(u[:,0])):
        u[i, 0] = f(delY * i + y[0])  #Left BCs
        u[i, -1] = g(delY * i + y[0]) #Right BCs
    for i in range(0, len(u[0,:])):
        u[0, i] = p(delX * i + x[0])  #Lower BCs
        u[-1, i] = r(delX * i + x[0]) #Upper BCs
    for i in range(stepsH+1):
        u[stepsY - i_h, i + j_h] = 100
    i = 0
    j = 0
    curRow = 0
    stepsX -= 2
    stepsY -= 2
    for i in range(0, stepsY + 1):
        for j in range(0, stepsX + 1):
            curRow = i*(stepsX + 1) + j
            A[curRow, curRow] = 4
            if(i == 0):#Bottom BC, p(x)
                b[curRow, 0] += p(delX *(j+1) + x[0])
            else:
                A[curRow, curRow - stepsX - 1] = -1

            if(i == stepsY):#Upper BC, r(x)
                b[curRow, 0] += r(delX *(j+1) + x[0])
            else:
                A[curRow, curRow + stepsX + 1] = -1

            if(j == 0):#Left bound is BC, f(y)
                b[curRow, 0] += f(delY *(i+1) + y[0])
            else:
                A[curRow, curRow - 1] = -1

            if(j == stepsX):#Right bound is BC, g(y)
                b[curRow, 0] += g(delY *(i+1) + y[0])
            else:
                A[curRow, curRow + 1] = -1

            if(i == stepsY - i_h + 1 and (j >= j_h-1 and j <= stepsX - j_h+1)):
                A[curRow, curRow + 1] = 0
                A[curRow, curRow - 1] = 0
                A[curRow, curRow + stepsX + 1] = 0
                A[curRow, curRow - stepsX - 1] = 0
                A[curRow, curRow] = 1
                b[curRow, 0] = 100
            elif((j >= j_h-1 and j <= stepsX - j_h+1) and (i >= i_h-1 and i <= stepsY - i_h+1)):
                A[curRow, curRow + 1] = 0
                A[curRow, curRow - 1] = 0
                A[curRow, curRow + stepsX + 1] = 0
                A[curRow, curRow - stepsX - 1] = 0
                b[curRow, 0] = 0

    temps = np.linalg.solve(A, b)
    for i in range(1, len(u[0, :]) - 1):
        u[i, 1:-1] = temps[(stepsX+1)*(i-1):((stepsX+1)*(i-1) + stepsX + 1), 0].T
    return x_points, y_points, u


L = [0, 10]
W = [0, 10]
fig1 = plt.figure(1, figsize=(12,7))
ax = plt.axes(projection = '3d')
x, y, u = CFD_Rectangle_Hole(L, W, f, g, p, r, 50, 50, 4)
X, Y = np.meshgrid(x, y)
ax.plot_surface(X, Y, u, cmap='gist_heat', alpha=0.8)
ax.set_title('Laplace\'s Eqn Solution x and y steps = 50')
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('U(x,y)')
plt.show()
```

Figure 38: Centered Finite Difference Method Code for Part 6