Alex Merino

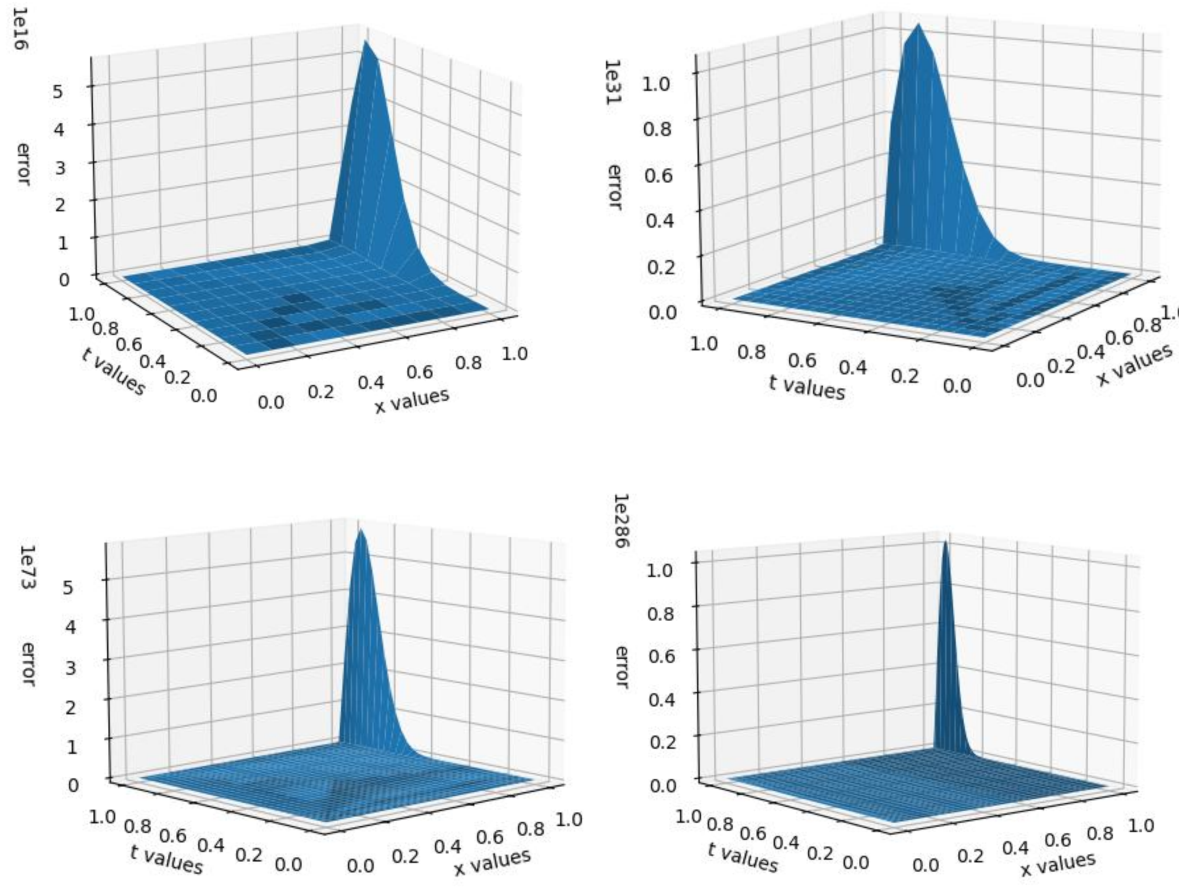
MTH 4311

Dr. Pei Liu

5/8/2025

Program 4

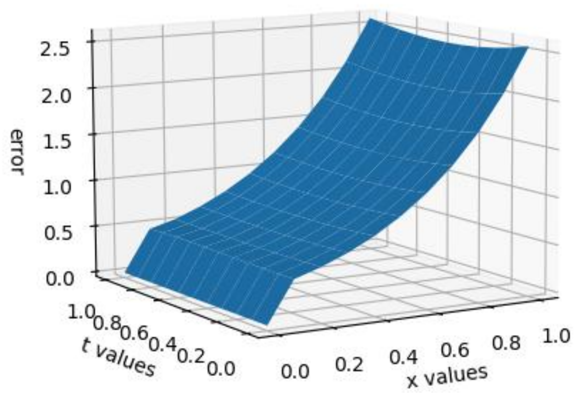
Question 1)

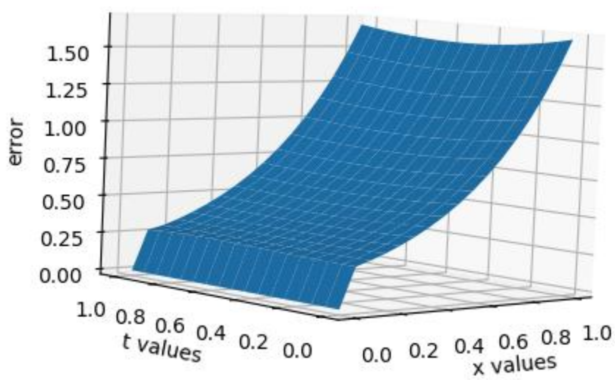


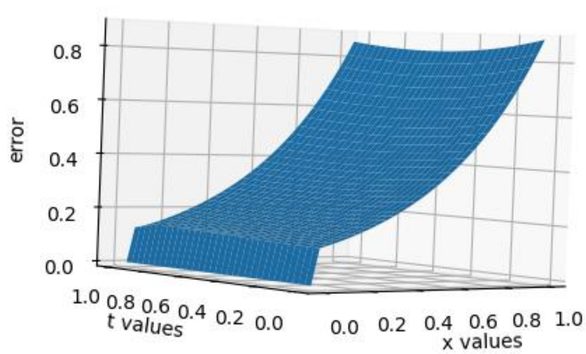
Above are four plots which show the error of the forward difference method given the PDE from question 1. The values of h and k are 0.1, 2^{-4}, 2^{-5}, and 0.01. For all x and t values, other than a x=1, the error between the actual and estimated plots is minimal. At x=1, the error skyrockets to values at 10^{16} and above for the smaller step sizes. With the larger errors at the end, we get the infinity norms, which are shown below. There is an inverse relationship between the norm and the predicted order of error, which is not what we should be seeing.

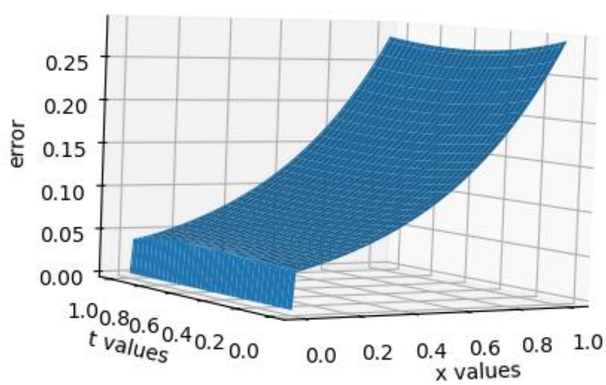
```
The infinity norm of the error is 5.6243148792366184e+16, h + k^2 is 0.11000000000000001
The infinity norm of the error is 1.0548527475811548e+31, h + k^2 is 0.06640625
The infinity norm of the error is 5.783312303834732e+73, h + k^2 is 0.0322265625
C:\Users\1738a\AppData\Local\Programs\Python\Python313\Lib\site-packages\numpy\linalg\_lir
  s = (x.conj() * x).real
The infinity norm of the error is 1.0284500277877905e+286, h + k^2 is 0.0101
[Finished in 5.2s]
```

Backwards Difference





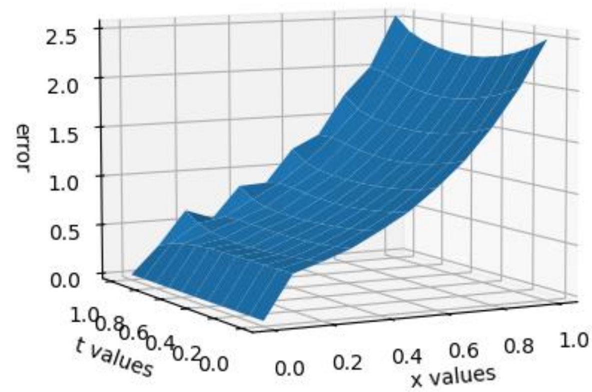


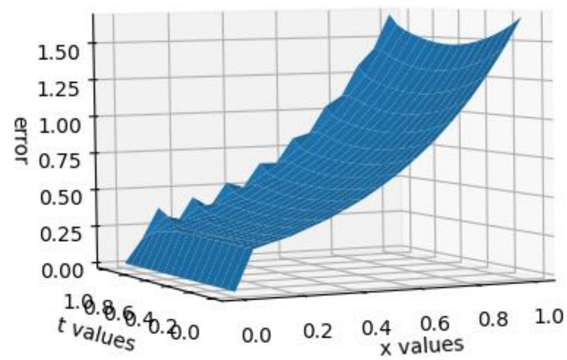


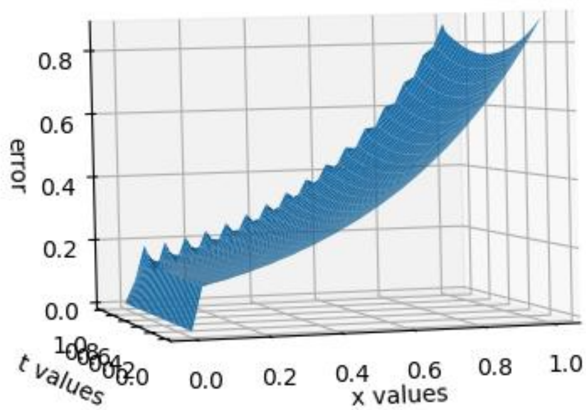
Using the same h and k values, we calculated the backwards difference method for heat equations. The graphs of the errors look much better with the error shrinking each decrease instead of increasing. The largest change in error is the first step, and after the change in error looks to be quadratic in the x and t directions, with it being much flatter in the t direction. The errors are shown below, with the order of them following our estimated order but at a slightly larger magnitude.

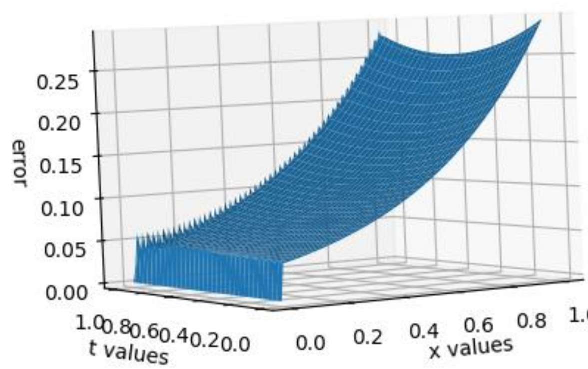
```
The infinity norm of the error is 1.5620370462549786, h + k^2 is 0.11000000000000001
The infinity norm of the error is 1.0138012985811073, h + k^2 is 0.06640625
The infinity norm of the error is 0.5240122911092051, h + k^2 is 0.0322265625
The infinity norm of the error is 0.17169472203506686, h + k^2 is 0.0101
[Finished in 129.4s]
```

Crank-Nicolson









Above, still with the same h and k values, is the error plots for Crank-Nicolson method. When compared to the backwards-difference method we still have the same magnitude of error, however, at t=1, there are some oscillations, which are displayed by the error not being smooth. These oscillations come from the method involving a matrix vector product before solving. The vector comes from the previous step which already has error from previous steps. Below is the infinity norms of the error vector. The error does not fully match the order predicted, but it is relatively close and still decreasing unlike in the forward difference.

```
The infinity norm of the error is 1.604083869146205, h^2 + k^2 is 0.020000000000000004
The infinity norm of the error is 1.0229854444737079, h^2 + k^2 is 0.0078125
The infinity norm of the error is 0.5209899104419016, h^2 + k^2 is 0.001953125
The infinity norm of the error is 0.1710688662967108, h^2 + k^2 is 0.0002
```

Question 2)

$$u(x,t) = \sin(\pi x)\cos(4\pi t)$$

$$u(0,t) = \sin(0)\cos(4\pi t) = 0 \text{ Correct}$$

x) $u(1,t) = \sin(\pi)\cos(4\pi t) = 0$ Correct

$$u(x,0) = \sin(\pi x)\cos(0) = \sin(\pi x) \text{ Correct}$$

$$u_t = -4\pi \sin(\pi x)\sin(4\pi t)$$

$$u_t(x,0) = -4\pi \sin(\pi x)\sin(4\pi 0) = 0 \text{ Correct}$$

$$u_{tt} = -16\pi^2 \sin(\pi x)\cos(4\pi t)$$

$$u_x = \pi \cos(\pi x)\cos(4\pi t)$$

$$u_{xx} = -\pi^2 \sin(\pi x)\cos(4\pi t)$$

Since they are the same, we can substitute

$$u_{tt} = 16 u_{xx}$$

Therefore proving that $u(x,t) = \sin(\pi x)\cos(4\pi t)$ is the exact solution

Above is the work which verifies that the equation given is the exact solution to the wave equation given. And below is the table which shows the numerical solution and error at (0.25, 0.75). The errors for all values of h and k are extremely similar to each other, with almost no difference between the smallest h and k values. Despite the errors being similar they are all relatively far away from the true value. This is because the forward difference method brings in a lot of error to the estimations and since we are near the end of the t values there has already been a lot of error brought in.

```
h       k       num. sol.       error
__________________________________________
0.06250 0.01562 0.5898759       1.29698
0.03125 0.00781 0.5882927       1.29540
0.01562 0.00391 0.5879743       1.29508
0.00781 0.00195 0.5879437       1.29505
0.00391 0.00098 0.5879392       1.29505
0.00195 0.00049 0.5879382       1.29504
0.00098 0.00024 0.5879379       1.29504
[Finished in 1.6s]
```

Question 3)

$$u(x,y) = x^2 + y^2$$

$u(x,1) = x^2 + 1^2 = x^2 + 1$ Correct

$u(x,2) = x^2 + 2^2 = x^2 + 4$ Correct

$u(3,y) = 3^2 + y^2 = y^2 + 9$ Correct

$u(5,y) = 5^2 + y^2 = y^2 + 25$ Correct

We know that

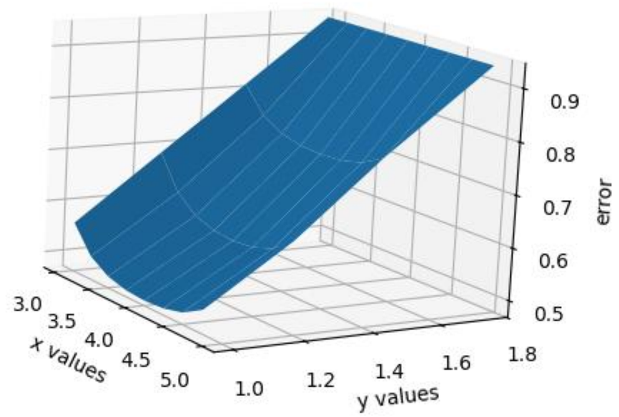
$$\Delta u + r(x,y)u = f(x,y)$$

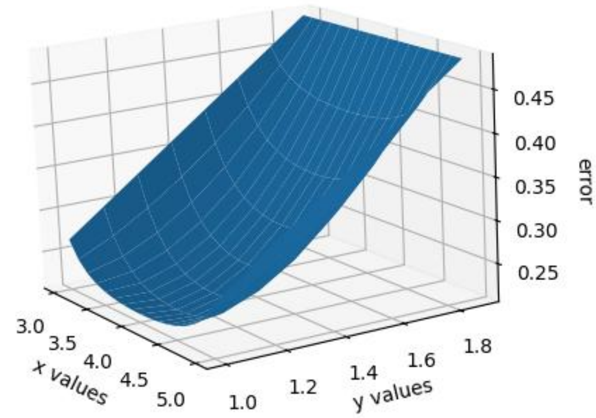
for Finite Element, plugging in our PDE we get

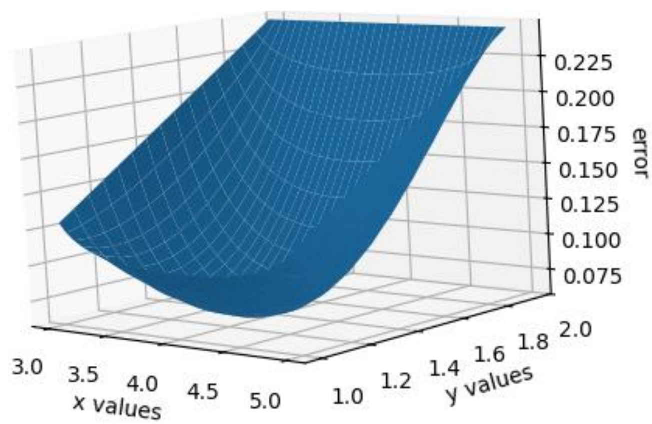
$$r(x,y) = \frac{1}{x^2+y^2}, \quad f(x,y) = 5$$

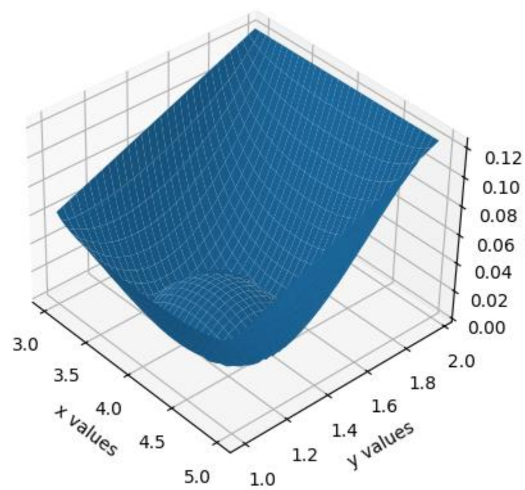
Above is the verification of the exact solution for the elliptic equation given. In this proof we also work out the function f(x, y) and r(x, y), which are necessary for the finite element and finite difference methods.

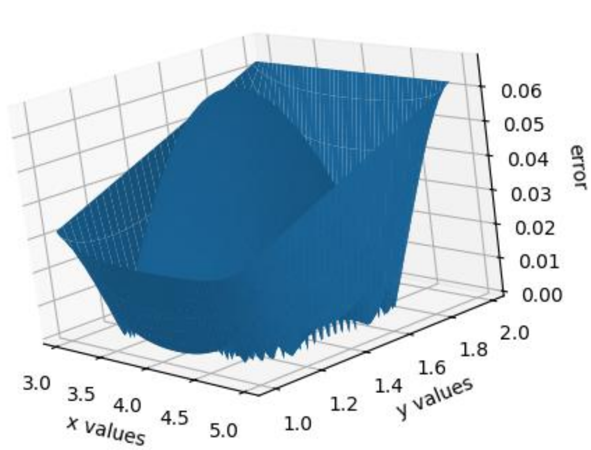
Finite Difference Method

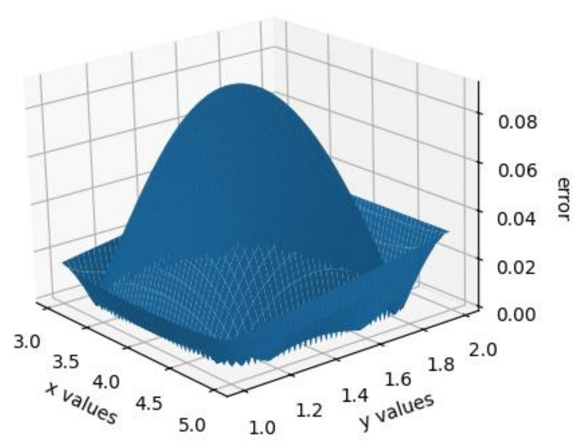








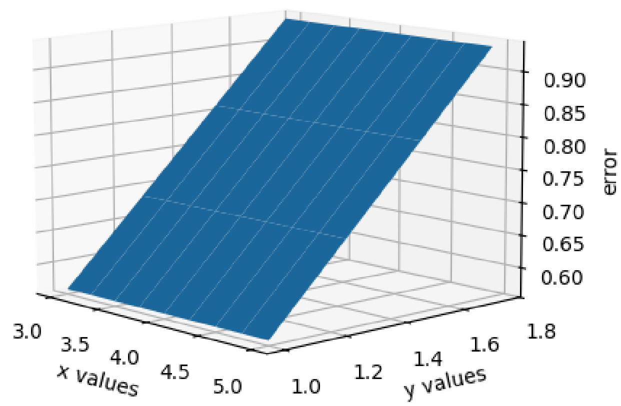


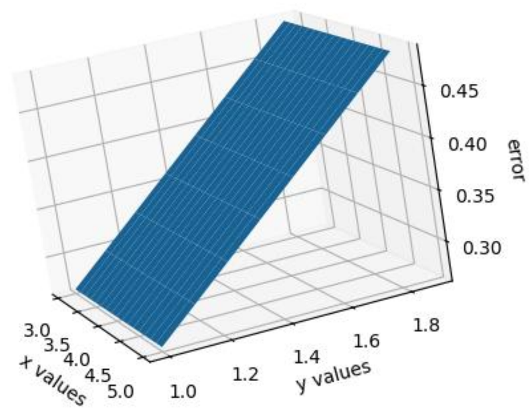


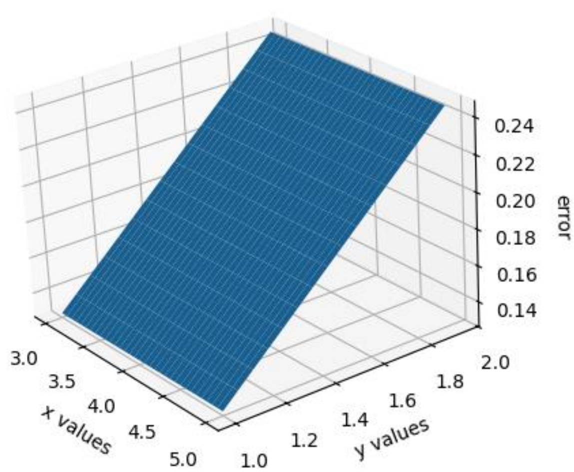
Above are the 6 error plots with decreasing step size. As the step size decreases the shape of the error plot changes dramatically where the largest error begins on the boundaries, then transitions into the center points. Since it is the finite difference method, there will always be some error introduced to the system, but it is interesting to see the pattern. Specifically at the borders, the error seems to decrease by a factor of 2 for each power of 2decrease in the step size. Since the error is in the order of h^2 this pattern makes sense. As for the time increase, it can be seen below, it is increasing exponentially, similarly to the exponential decrease in the step size. These times were calculated from beginning of the method to the end, it is also noted that the solver used a sparse matrix.

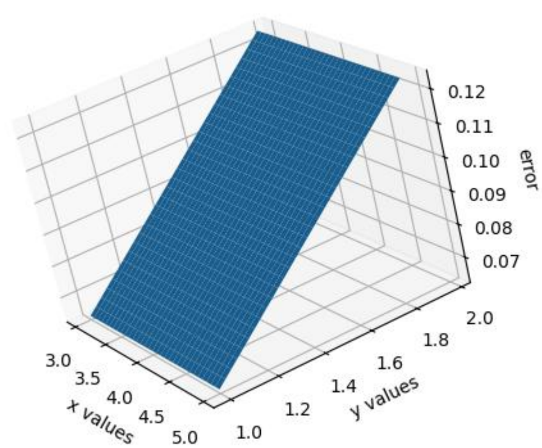
```
Time to solve with h_x = h_y = 0.25 is 0.0004secs
Time to solve with h_x = h_y = 0.125 is 0.0007secs
Time to solve with h_x = h_y = 0.0625 is 0.0046secs
Time to solve with h_x = h_y = 0.03125 is 0.0377secs
Time to solve with h_x = h_y = 0.015625 is 0.4836secs
Time to solve with h_x = h_y = 0.0078125 is 7.3680secs
```

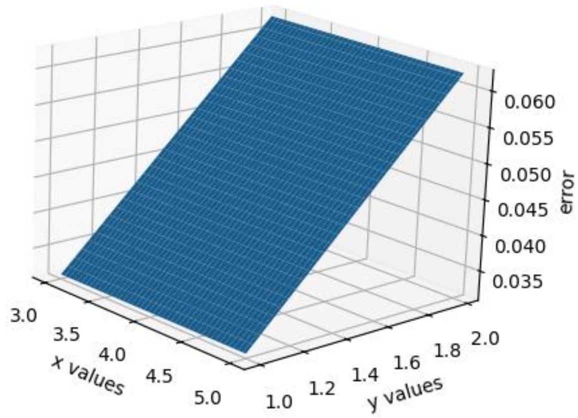
Finite Element Method

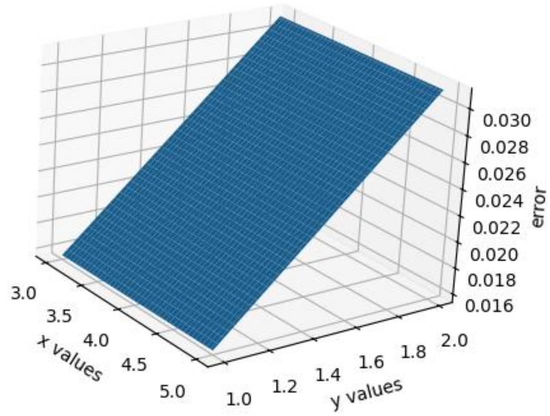












Above are the 6 error plots for the 6 different h values of the finite element method. As the h values get smaller, the error decays by one half. Our h values are decreasing by a power of two, which corresponds to the order of the error decreasing by a factor of two, since it is the step size raised to the second power. Below is the time it took for each calculation to run. The time to run gets exponentially larger as the h values get exponentially smaller. These times were calculated from beginning of the method to the end, it is also noted that the solver used a sparse matrix.

```
Time to solve with h_x = h_y = 0.25 is 0.0005secs
Time to solve with h_x = h_y = 0.125 is 0.0015secs
Time to solve with h_x = h_y = 0.0625 is 0.0058secs
Time to solve with h_x = h_y = 0.03125 is 0.0464secs
Time to solve with h_x = h_y = 0.015625 is 0.5379secs
Time to solve with h_x = h_y = 0.0078125 is 6.6599secs
```