Numerical Techniques Laboratory

Assignment Gauss Seidel | Tanishq Jasoria | 16MA20047

Gauss Seidel Iterative method

In [10]:

```
import numpy as np
from matplotlib import pyplot as plt
from mpl_toolkits import mplot3d
from copy import copy

%matplotlib inline
plt.rcParams['figure.figsize'] = [10, 10]
```

Scheme

At any $k+1^{th}$ iteration we can write the difference equation as

$$u_{ij}^{k+1} = \frac{\partial y^2(u_{i-1j}^{k+1} + u_{i+1j}^{k}) + \partial x^2(u_{ij-1}^{k+1} + u_{ij+1}^{k})}{2(\partial x^2 + \partial y^2)}$$

And continue the iteration till

$$\max_{\substack{1\le i\le N-1\\ 1\le j\le M-1}} \left|u_{ij}^{k+1} - u_{ij}^{k}\right| \le \epsilon$$

We need to solve the PDE

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

for the boundary conditions

$$0 \le x\ , y \le 4$$

$u = x^2y^2$ on the boundaries

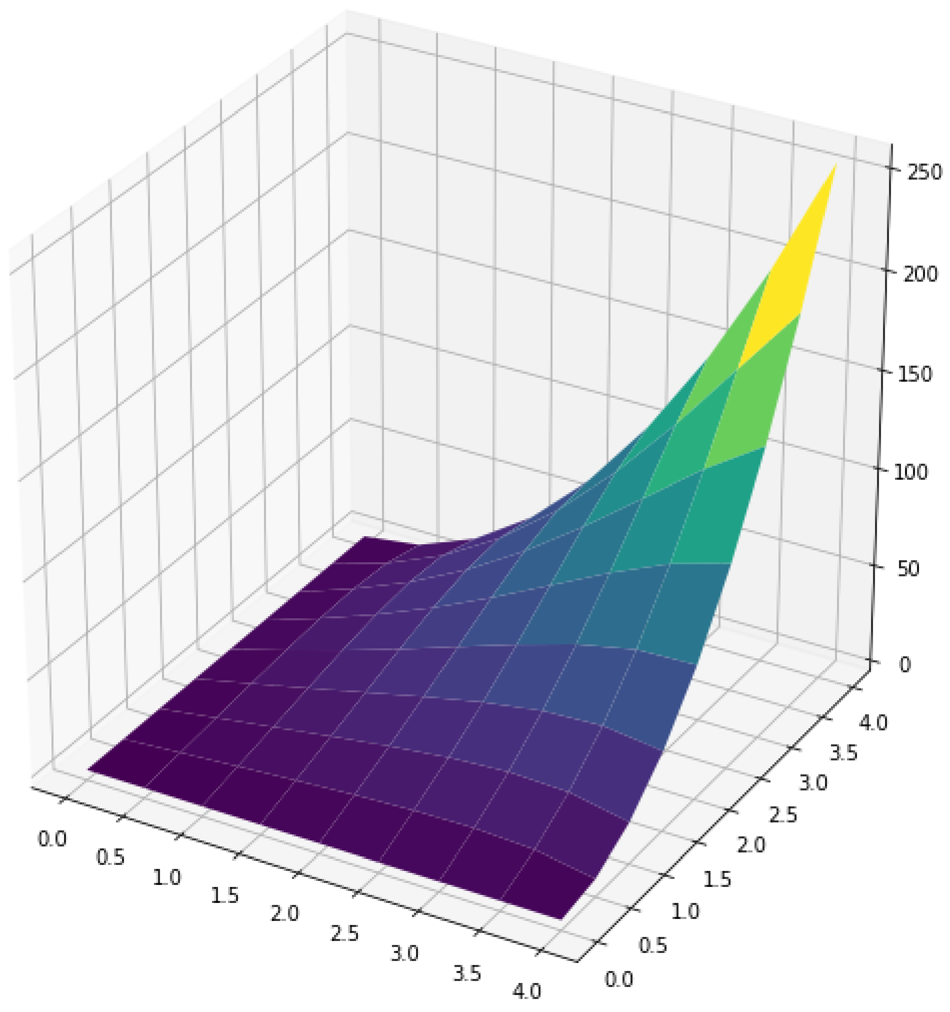
In [11]:

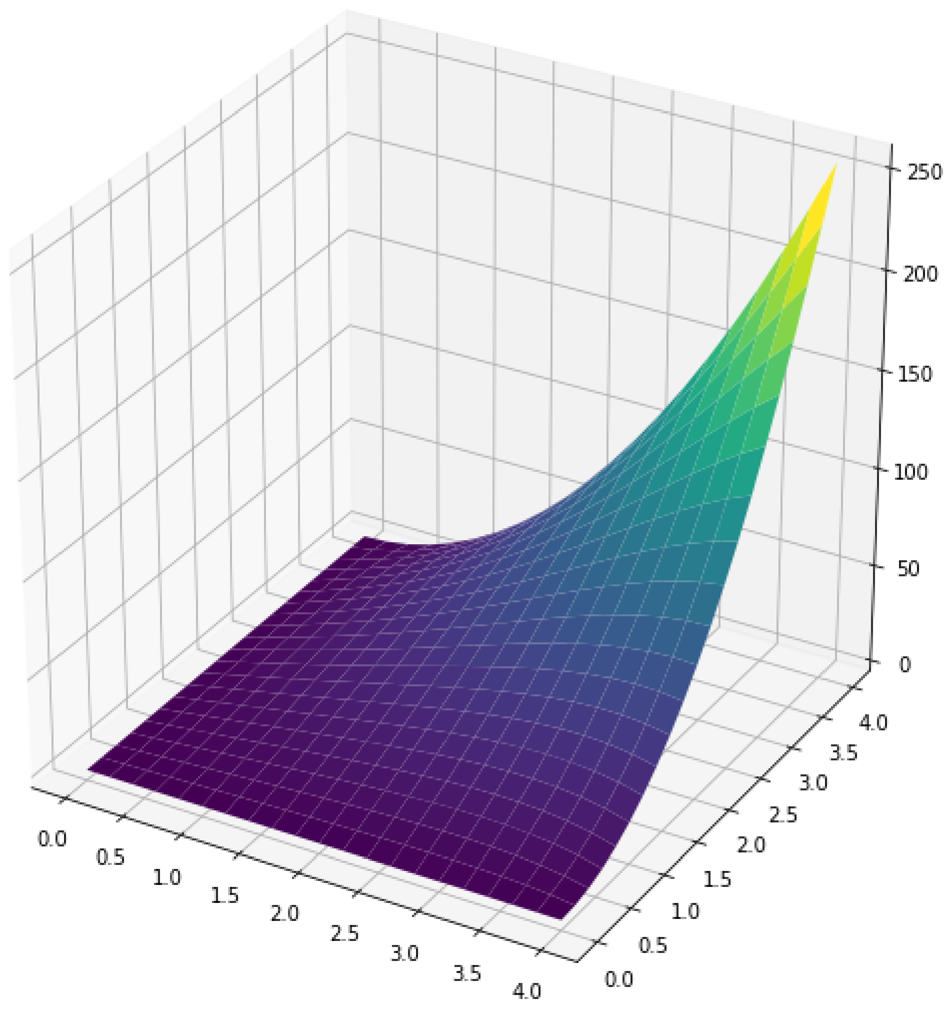
```
def GaussSeidel(u, space_step_x, space_step_y, epsilon):
    u_ = copy(u)
    steps_x , steps_y = int(u.shape[0]) - 1, int(u.shape[1]) - 1
    u_iter = np.zeros((steps_x + 1, steps_y + 1))
    difference_matrix = np.ones((steps_x, steps_y))
    iterations = 0
    while(np.amax(difference_matrix)>epsilon):
        iterations += 1
        for i in range(steps_x + 1):
            for j in range(steps_y + 1):
                if( i == 0 or i == steps_x or j == 0 or j == steps_y):
                    u_iter[i][j] = u[i][j]
                else:
                    u_iter[i][j] = (space_step_y**2*(u_iter[i-1][j] + u[i+1][j]) +
                                    space_step_x**2*(u_iter[i][j-1] + u[i][j+1]))/(

#                     print((space_step_y**2*(u_iter[i-1][j] + u[i+1][j]) + space_s
        difference_matrix = np.absolute(u_iter - u)
        u = copy(u_iter)
    print("Iterations: ", iterations)

    return u
```

In [12]:

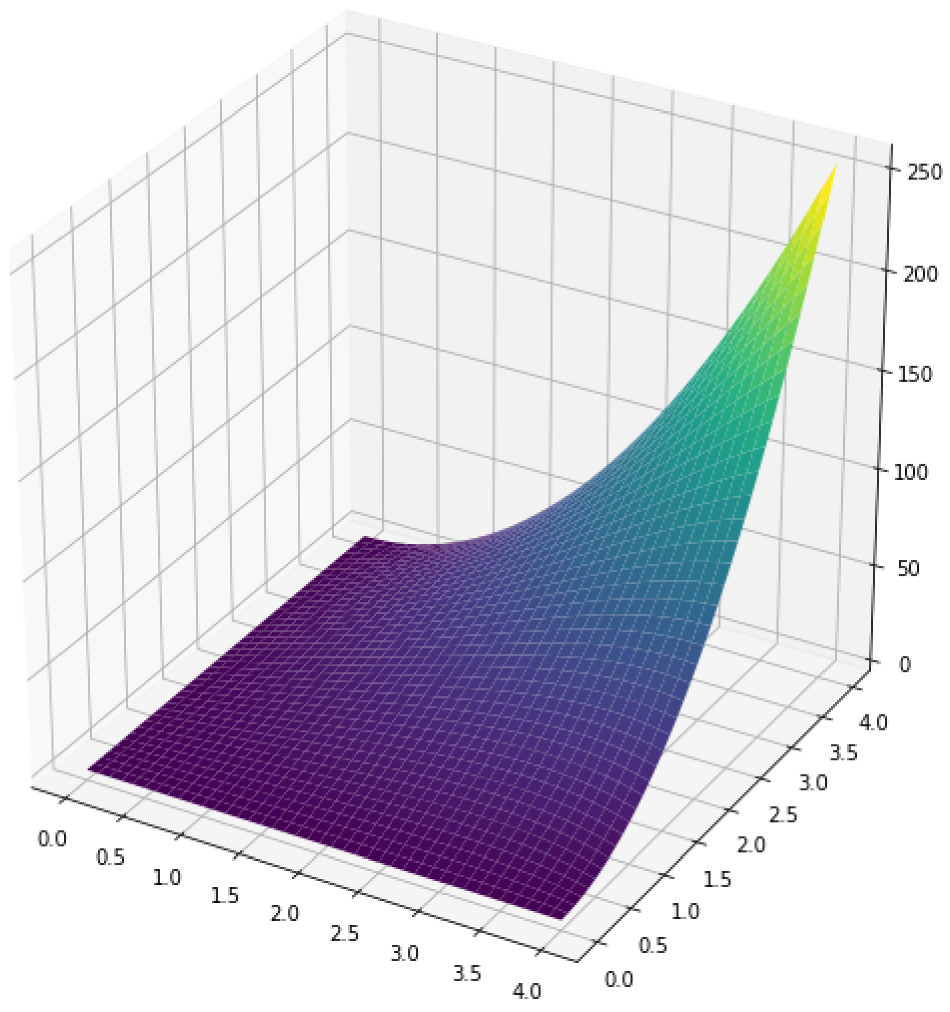
```
space_step_x = space_step_y = [0.5, 0.2, 0.1]
x0 = y0 = 0
xn = yn = 4
def initialize(x0, y0, steps_x, steps_y, space_step_x, space_step_y ):
    #     initialize the solution wrt u = x^2*y^2
    u = np.zeros((int(steps_x+1), int(steps_y+1)))
    for i in range(int(steps_x + 1)):
        for j in range(int(steps_y + 1)):
            if( i == 0 or i == steps_x or j == 0 or j == steps_y):
                u[i][j] = (x0 + i*space_step_x)**2 * (y0 + j*space_step_y)**2
            else:
                u[i][j] = 0
    return u
for steps in space_step_x:
    steps_x = steps_y = np.floor((xn - x0)/steps)
    x = np.linspace(x0, xn, num=int(steps_x + 1))
    y = np.linspace(y0, yn, num=int(steps_y + 1))

    u = initialize(x0, y0, steps_x, steps_y, steps, steps)
    u = GaussSeidel(u, steps, steps, epsilon=0.00001)
    fig = plt.figure()
    ax = plt.axes(projection='3d')
    X, Y = np.meshgrid(x,y)

    ax.plot_surface(X, Y, u, cmap='viridis')
```

```
Iterations:  90
Iterations:  489
Iterations:  1722
```


0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

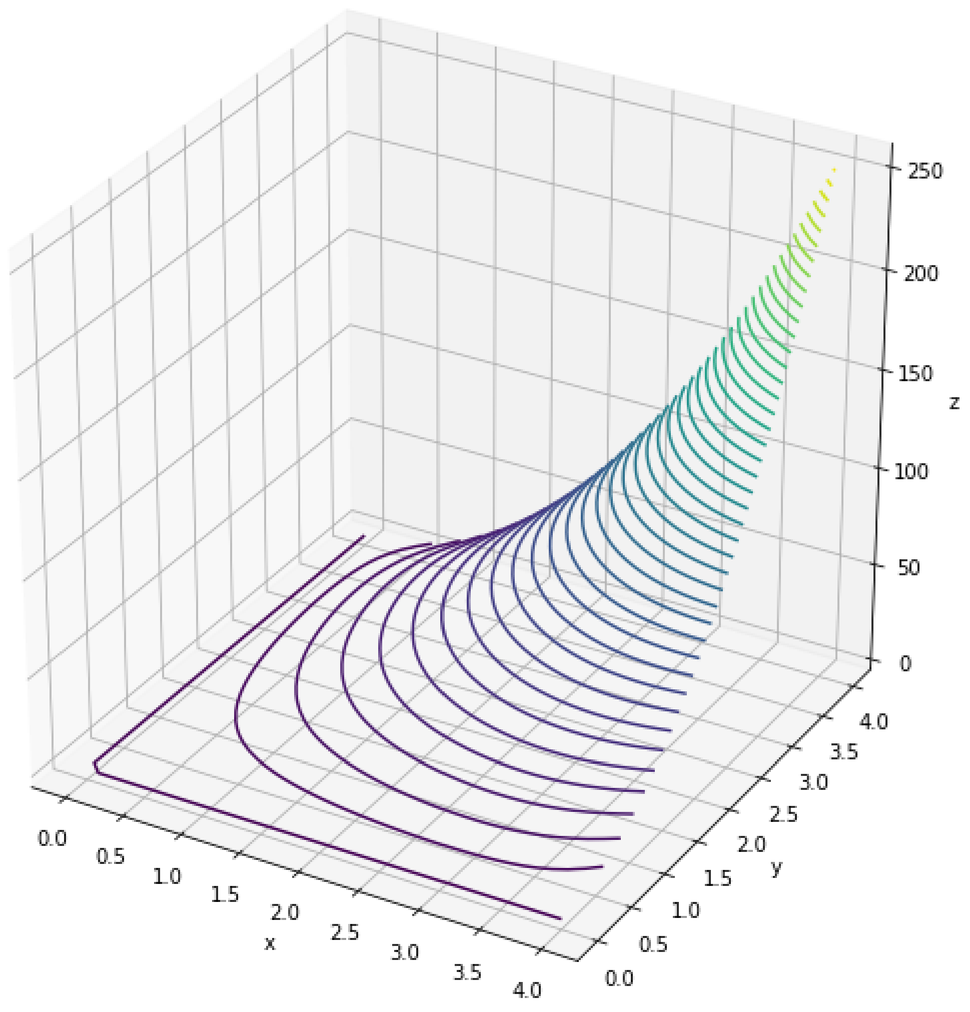


250
200
150
100
50
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0


0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
0
50
100
150
200
250

In [13]:

```
fig = plt.figure()
ax = plt.axes(projection='3d')
ax.contour3D(X, Y, u, 50, cmap='viridis')
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('z');
```



0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
x
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
y
0
50
100
150
200
250
z