In [6]:
```
## Assignment 1 Problems done with Python NumPy Package
import numpy as np
```

In [7]:
```
## Problem 3c
## Creating the matrix B as a 3x3
B = np.array([-13, -8, -4, 12, 7, 4, 24, 16, 7])
B = B.reshape((3,3))
print(B)
```

```
[[-13  -8  -4]
 [ 12   7   4]
 [ 24  16   7]]
```

In [8]:
```
## To get eig function need to import linalg then can get eigenvalues and vectors using
from numpy import linalg as LA
D, V = LA.eig(B)
print(D)
print(V)
```

```
[-1.  3. -1.]
[[-0.51214752 -0.40824829 -0.02464807]
 [ 0.38411064  0.40824829 -0.41725537]
 [ 0.76822128  0.81649658  0.90845497]]
```

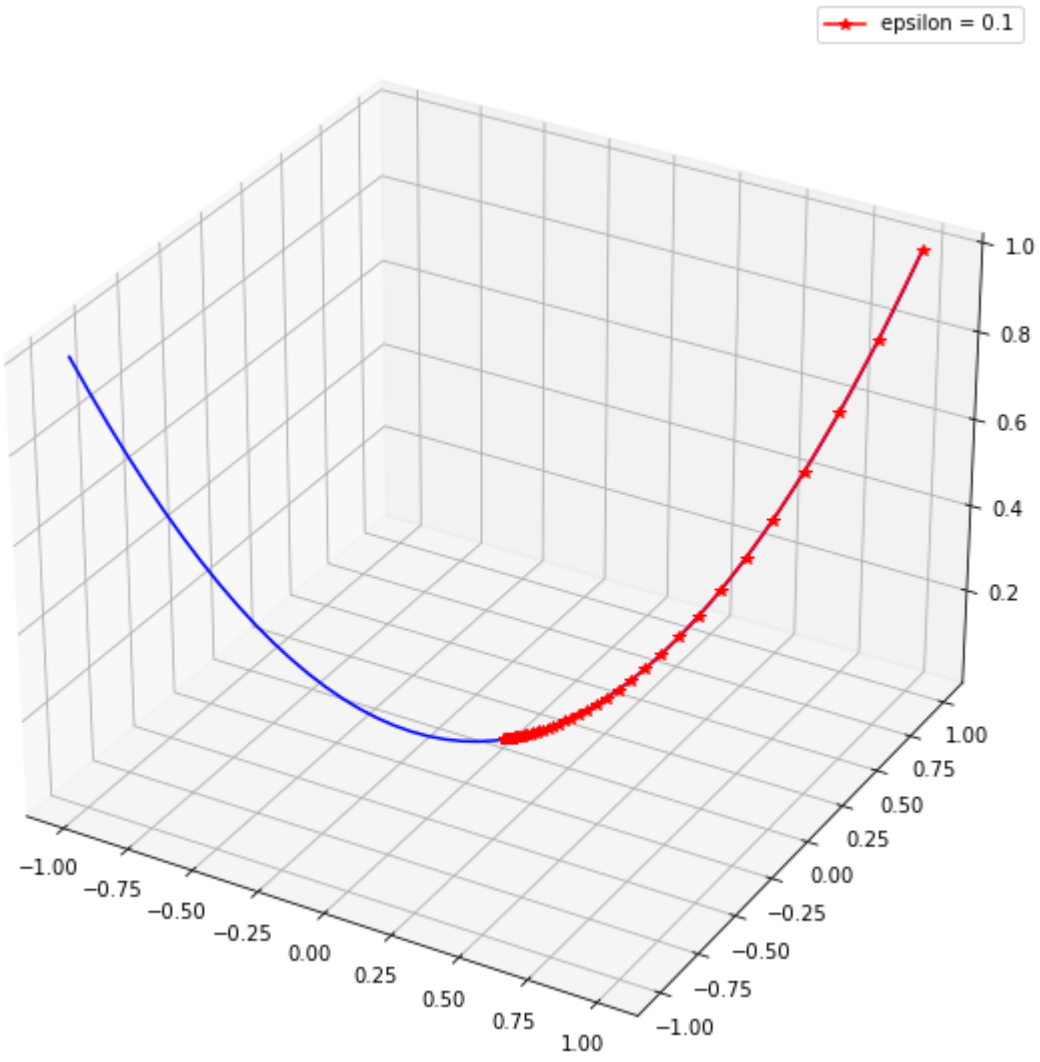
In [15]:
```
## Problem 5 B: Implementing Gradient Method
## Function: F(x) = x1^2 + x2^2 -x1x2

import numpy as np
import math

## Defines Function
def func(x):
    return x[0]**2 + x[1]**2 - x[0]*x[1]

## Gradient of function
def grad(x):
    dx1 = 2*x[0] - x[1]
    dx2 = 2*x[1] - x[0]
    return np.array([dx1, dx2])

## Gradient method defined
def grad_method(xit, stepSize, epsilon, max_iter, past_x):
    """
      grad: gradient of function
       xit: intitial point
  stepSize: helps increment the gradient
   epsilon: stopping condition
  max_iter: maximum number of iterations if stopping condition is not met
    past_x: tracks past positions of x
    return: x position will update until convergence
    """
    ## Initialization to start cacl. in GM
    print(f"set {xit} as the starting point")

    for i in range(max_iter):
        ## Records iterations of x
        past_x.append(xit)
```

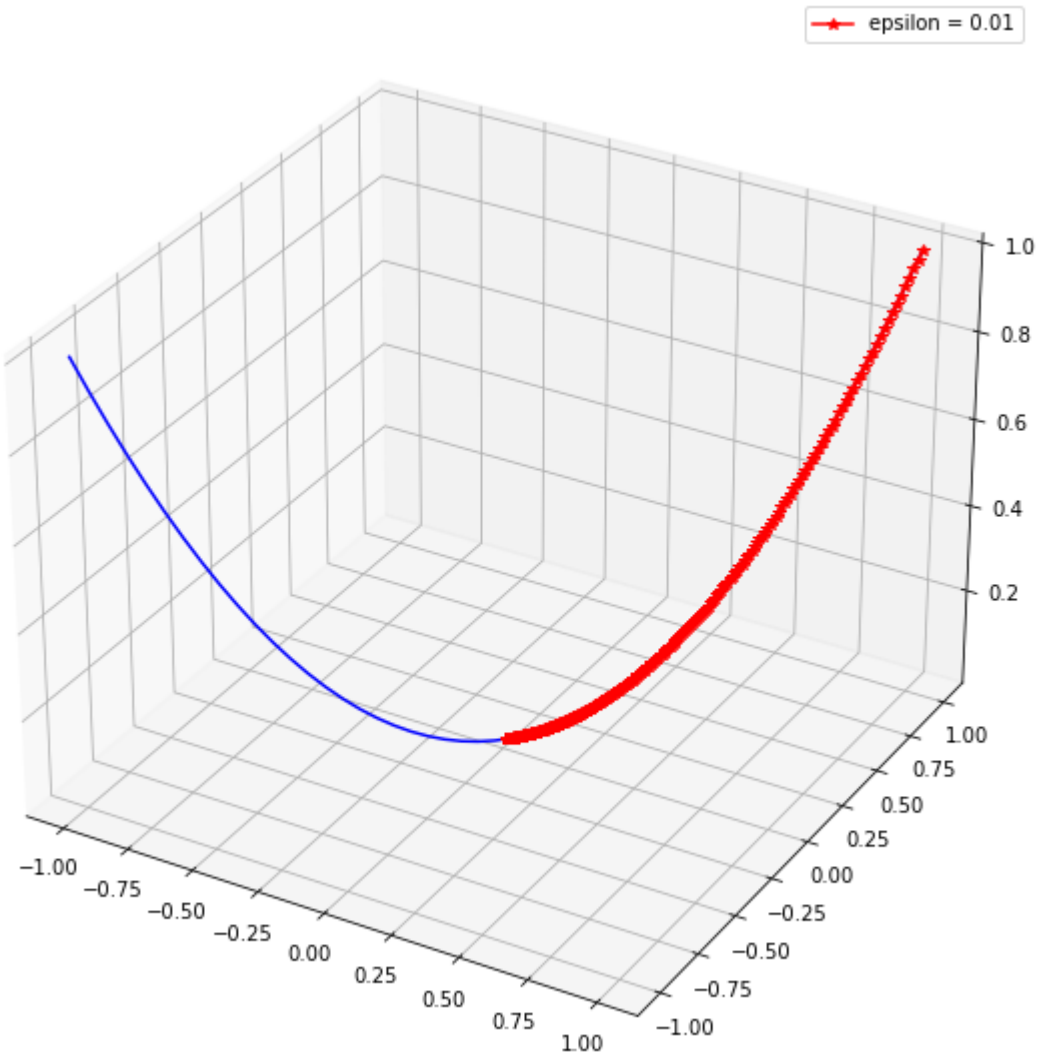
```
        prev_x = xit

        ## GM updating x
        xit = xit - stepSize*grad(xit)

        ## Stopping Condition
        if abs(func(prev_x)-func(xit)) < epsilon:
            print("The", i, "iteration ", xit)
            break



    print("[x1,x2] =", xit)
    print("Local minimum is: ",func(xit))
    print("\n")

    return past_x


Init_1 = grad_method(xit = np.array([1,1]), stepSize = 0.1, epsilon = 0.000001, max_ite
Init_2 = grad_method(xit = np.array([1,1]), stepSize = 0.01, epsilon = 0.000001, max_it
Init_3 = grad_method(xit = np.array([1,1]), stepSize = 0.001, epsilon = 0.000001, max_i


################################################################################
## Plotting GM and function
from matplotlib import pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

## Initializes values for function
xd = np.linspace(-1,1)
yd = np.linspace(-1,1)
zd = xd ** 2 + yd ** 2 - xd*yd

## Init_1
x0 = np.array(Init_1)[:,0]
x1 = np.array(Init_1)[:,1]
z = x0**2 + x1**2 - x0*x1

## Init_2
x0_2 = np.array(Init_2)[:,0]
x1_2 = np.array(Init_2)[:,1]
z_2 = x0_2**2 + x1_2**2 - x0_2*x1_2

## Init_3
x0_3 = np.array(Init_3)[:,0]
x1_3 = np.array(Init_3)[:,1]
z_3 = x0_3**2 + x1_3**2 - x0_3*x1_3


################################################################################
# Init_1: Epsilon = 0.1

fig=plt.figure(figsize = (10,10))
## Plotting objective function
ax = plt.axes(projection = '3d')
ax.plot3D(xd,yd,zd,color = 'b')

## Plotting GM data
ax.plot3D(x0,x1,z, color = 'r', marker = '*', label = 'epsilon = 0.1')
ax.legend()
```

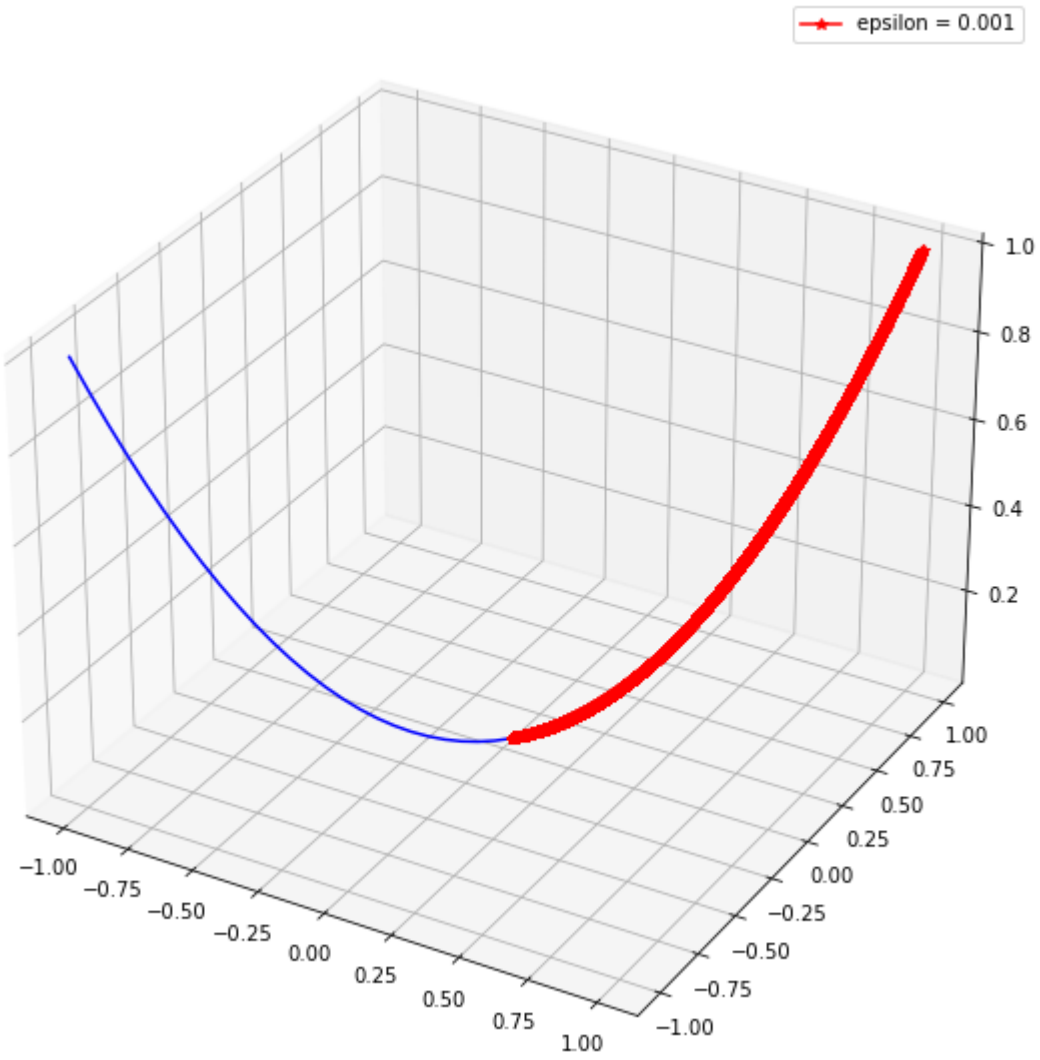
```
###############################################################################
# Init_2: Epsilon = 0.01

fig=plt.figure(figsize = (10,10))
## Plotting objective function
ax = plt.axes(projection = '3d')
ax.plot3D(xd,yd,zd,color = 'b')

## Plotting GM data
ax.plot3D(x0_2,x1_2,z_2, color = 'r', marker = '*', label = 'epsilon = 0.01')
ax.legend()
###############################################################################
# Init_3: Epsilon = 0.001

fig=plt.figure(figsize = (10,10))
## Plotting objective function
ax = plt.axes(projection = '3d')
ax.plot3D(xd,yd,zd,color = 'b')

## Plotting GM data
ax.plot3D(x0_3,x1_3,z_3, color = 'r', marker = '*', label = 'epsilon = 0.001')
ax.legend()
```

```
set [1 1] as the starting point
The 58 iteration  [0.00199668 0.00199668]
[x1,x2] = [0.00199668 0.00199668]
local minimum  3.9867234790105596e-06


set [1 1] as the starting point
The 493 iteration  [0.00697889 0.00697889]
[x1,x2] = [0.00697889 0.00697889]
local minimum  4.8704855856885694e-05


set [1 1] as the starting point
The 3799 iteration  [0.02232828 0.02232828]
[x1,x2] = [0.02232828 0.02232828]
local minimum  0.0004985520627322681
```

Out[15]: <matplotlib.legend.Legend at 0x18c00f5db80>


epsilon = 0.1
1.0
0.8
0.6
0.4
0.2
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00


epsilon = 0.01
1.0
0.8
0.6
0.4
0.2
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00


epsilon = 0.001
1.0
0.8
0.6
0.4
0.2
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00

In [ ]: