HW 3 : Convex Optimisation

Gradient Descent

In [1]:

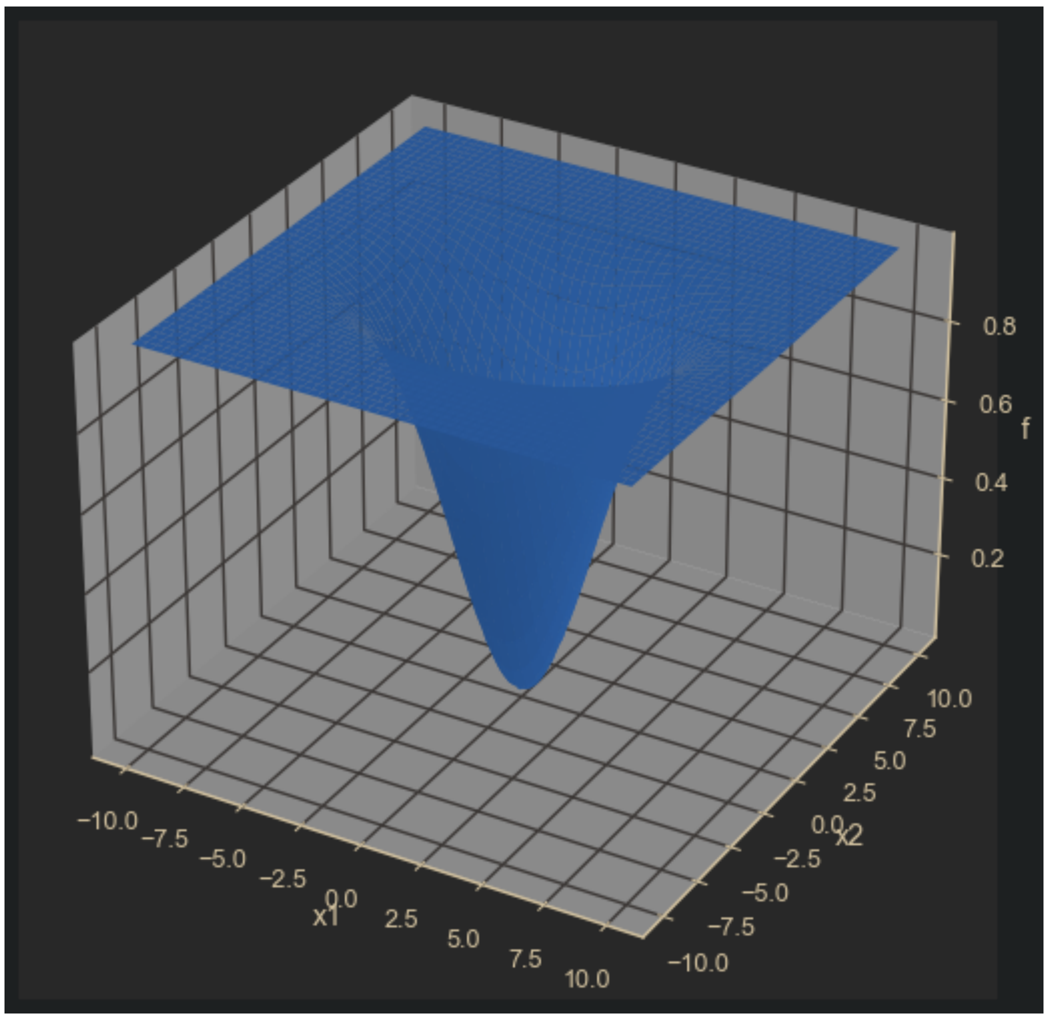
```
# Import all the required libraries
import numpy as np
import matplotlib.pyplot as plt
from jupyterthemes import jtplot
jtplot.style(theme='gruvboxd', context='notebook', ticks=True, grid=False)
```

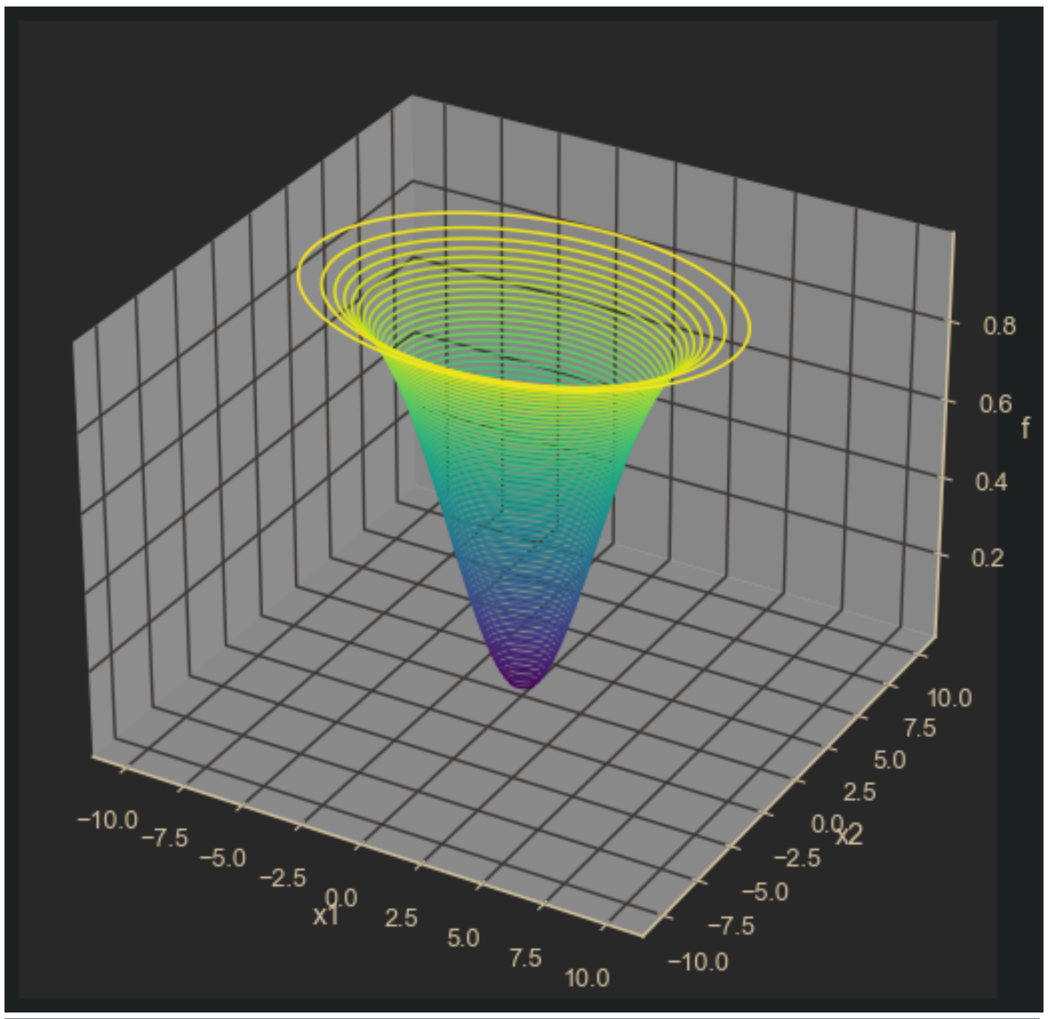
Steepest Descent (without line search)

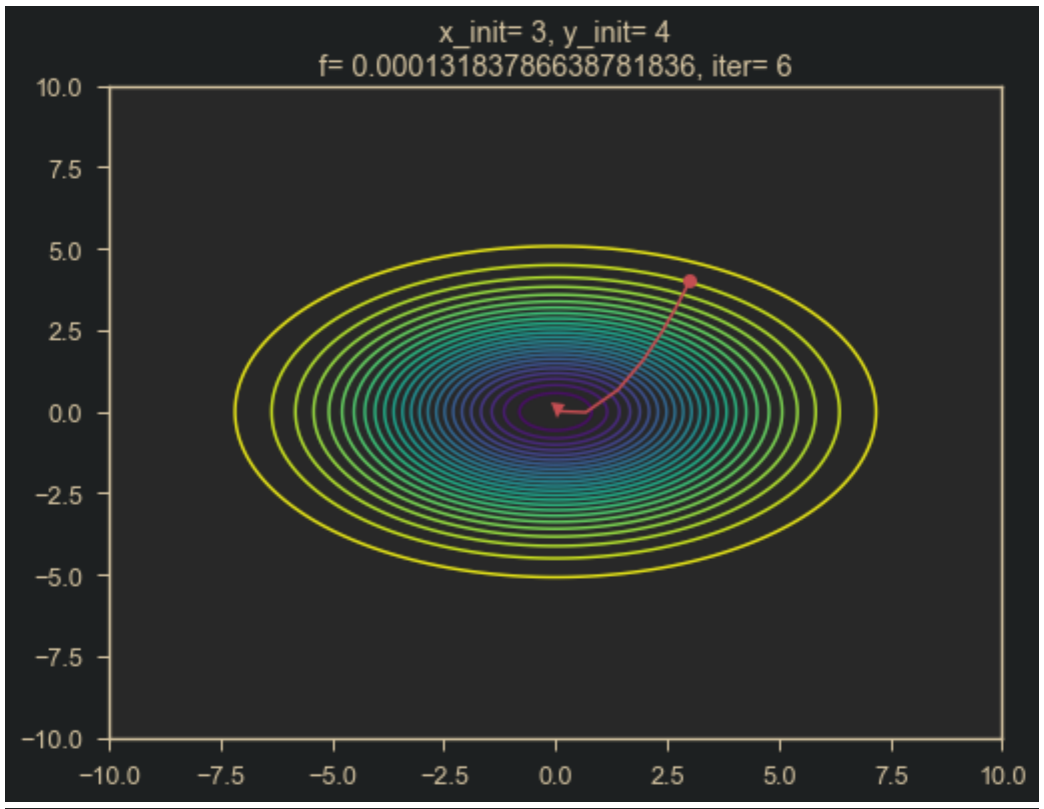
In [2]:

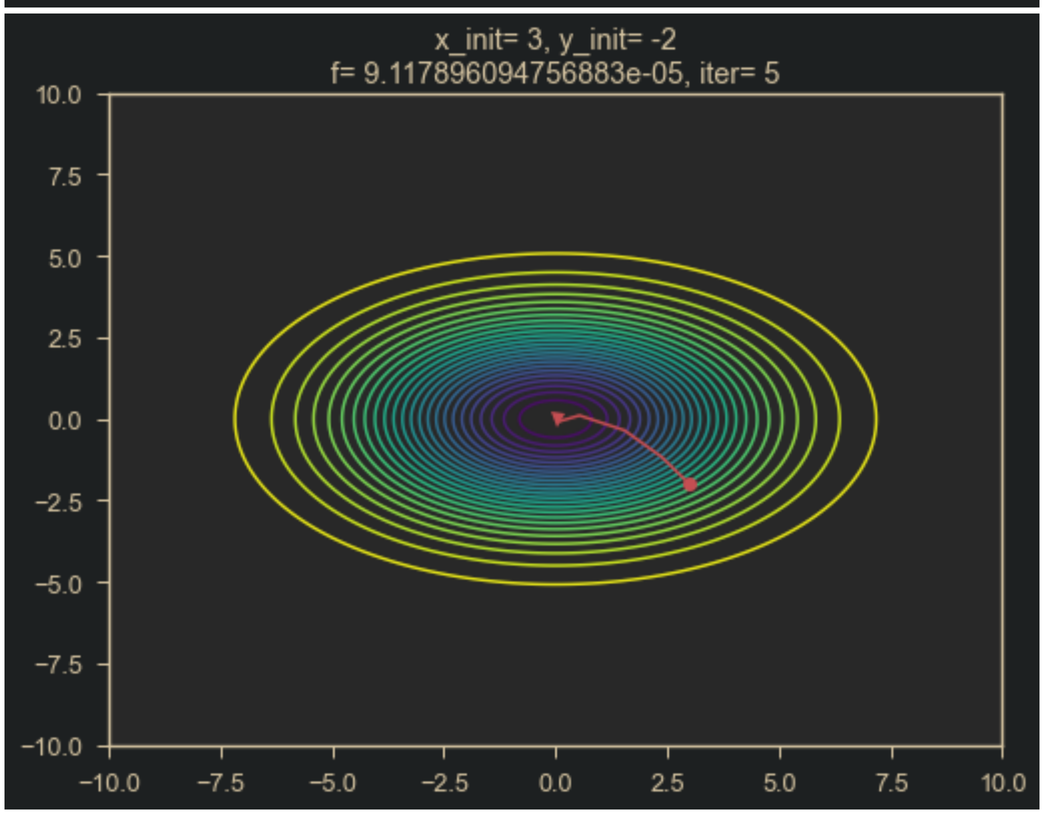
```python
# defining f
def f(x,y,sig_x,sig_y,std_x,std_y):
    return 1-np.exp(-((x-sig_x)**2/(2*std_x)+(y-sig_y)**2/(2*std_y)));
# defining g
def g_x (x,y,sig_x,sig_y,std_x,std_y):
    return f(x,y,sig_x,sig_y,std_x,std_y)*((x-sig_x)/std_x);
def g_y (x,y,sig_x,sig_y,std_x,std_y):
    return f(x,y,sig_x,sig_y,std_x,std_y)*((y-sig_y)/std_y);
# defining H
def H_xx (x,y,sig_x,sig_y,std_x,std_y):
    return f(x,y,sig_x,sig_y,std_x,std_y)*(1/std_x+((x-sig_x)/std_x)**2);
def H_yy (x,y,sig_x,sig_y,std_x,std_y):
    return f(x,y,sig_x,sig_y,std_x,std_y)*(1/std_y+((y-sig_y)/std_y)**2);
def H_xy (x,y,sig_x,sig_y,std_x,std_y):
    return f(x,y,sig_x,sig_y,std_x,std_y)*((y-sig_y)/std_y)*((x-sig_x)/std_x);
# alpha_k
def alpha (dk,H):
    return (dk.T@dk)/(dk.T@H@dk);
# defiing parameters for f (X)
sig_x = 0
sig_y = 0
std_x = 8
std_y = 4
# Generating x and y, 1000 data points from -10 to 10
x = np.linspace(-10,10,1000);
y = np.linspace(-10,10,1000);
# meshgrid for plotting 3D plots
X,Y = np.meshgrid(x,y);
# generating f(x,y)
fxy = f(X,Y,sig_x,sig_y,std_x,std_y);
# plotting Surface plot
plt.figure(figsize=(9,9))
ax = plt.axes(projection='3d')
ax.plot_surface(X, Y, fxy)
ax.set_xlabel('x1');
ax.set_ylabel('x2');
ax.set_zlabel('f');
#plotting contour 3D plot
plt.figure(figsize=(9,9))
ax = plt.axes(projection='3d')
ax.contour3D(X, Y, fxy,80)
ax.set_xlabel('x1');
ax.set_ylabel('x2');
ax.set_zlabel('f');
# array of initial values of x ,y
x_init_arr = [3,-7];
y_init_arr = [4,-2];
# looping over inital points
for x_init in x_init_arr:
    for y_init in y_init_arr:
        # counting iterations
        iter = 0;
        x_hist = np.array([]); # history array to store all values of x in each update
        y_hist = np.array([]); # history array to store all values of y in each update
        x_hist = np.append(x_hist,x_init); # appending initial value to array
        y_hist = np.append(y_hist,y_init); # appending initial value to array
        # loop to get updates till convergence
        while (g_x(x_hist[-1],y_hist[-1],sig_x,sig_y,std_x,std_y)>= 0.00001 or
               g_x(x_hist[-1],y_hist[-1],sig_x,sig_y,std_x,std_y)<=-0.00001 or
```

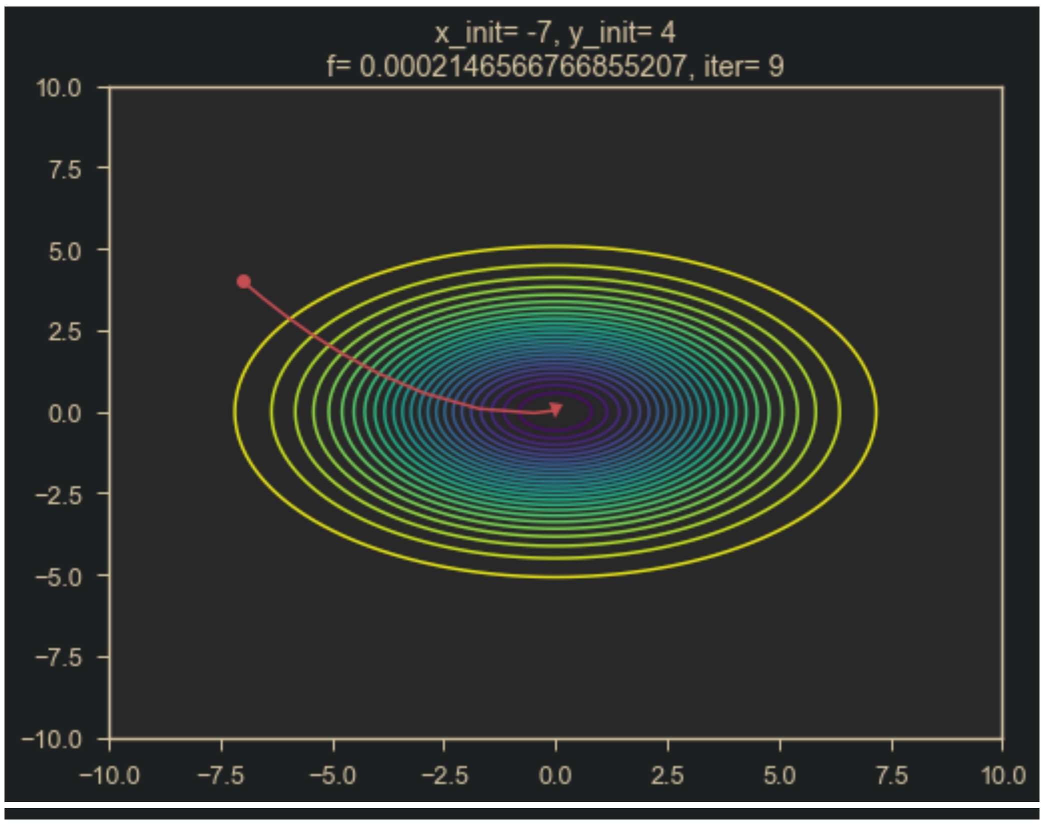
```
            dk = np.array([[-g_x(x_old,y_old,sig_x,sig_y,std_x,std_y)],
[-g_y(x_old,y_old,sig_x,sig_y,std_x,std_y)]]);
            Hk =
np.array([[H_xx(x_old,y_old,sig_x,sig_y,std_x,std_y),H_xy(x_old,y_old,sig_x,sig_y,std_x,std_y)],

[H_xy(x_old,y_old,sig_x,sig_y,std_x,std_y),H_yy(x_old,y_old,sig_x,sig_y,std_x,std_y)]]);
            alpha_k = alpha(dk,Hk);
            x_up = x_old + alpha_k*dk[0]; # updating x
            y_up = y_old + alpha_k*dk[1]; # updating y
            x_hist = np.append(x_hist,x_up); # appending updated x to history array
            y_hist = np.append(y_hist,y_up); # appending updated y to history array
            iter+=1;
        # defining figure and size
        plt.figure(figsize=(8,6))
        # plotting 2D contour plots
        plt.contour(X, Y, fxy,30);
        # plotting history of updates
        plt.plot(x_hist,y_hist,'r')
        # plotting optimal value
        plt.plot(x_hist[-1],y_hist[-1],color='r',marker='v')
        # plotting initial value
        plt.plot(x_hist[0],y_hist[0],color='r',marker='o')
        # title
        plt.title("x_init= "+str(x_init)+", y_init= "+str(y_init)
                  +"\nf= "+str(f(x_hist[-1],y_hist[-1],sig_x,sig_y,std_x,std_y))+", iter=
"+str(iter))
```




f
x1
x2
0.8
0.6
0.4
0.2
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0


x_init= 3, y_init= 4
f= 0.00013183786638781836, iter= 6


x_init= 3, y_init= -2
f= 9.117896094756883e-05, iter= 5



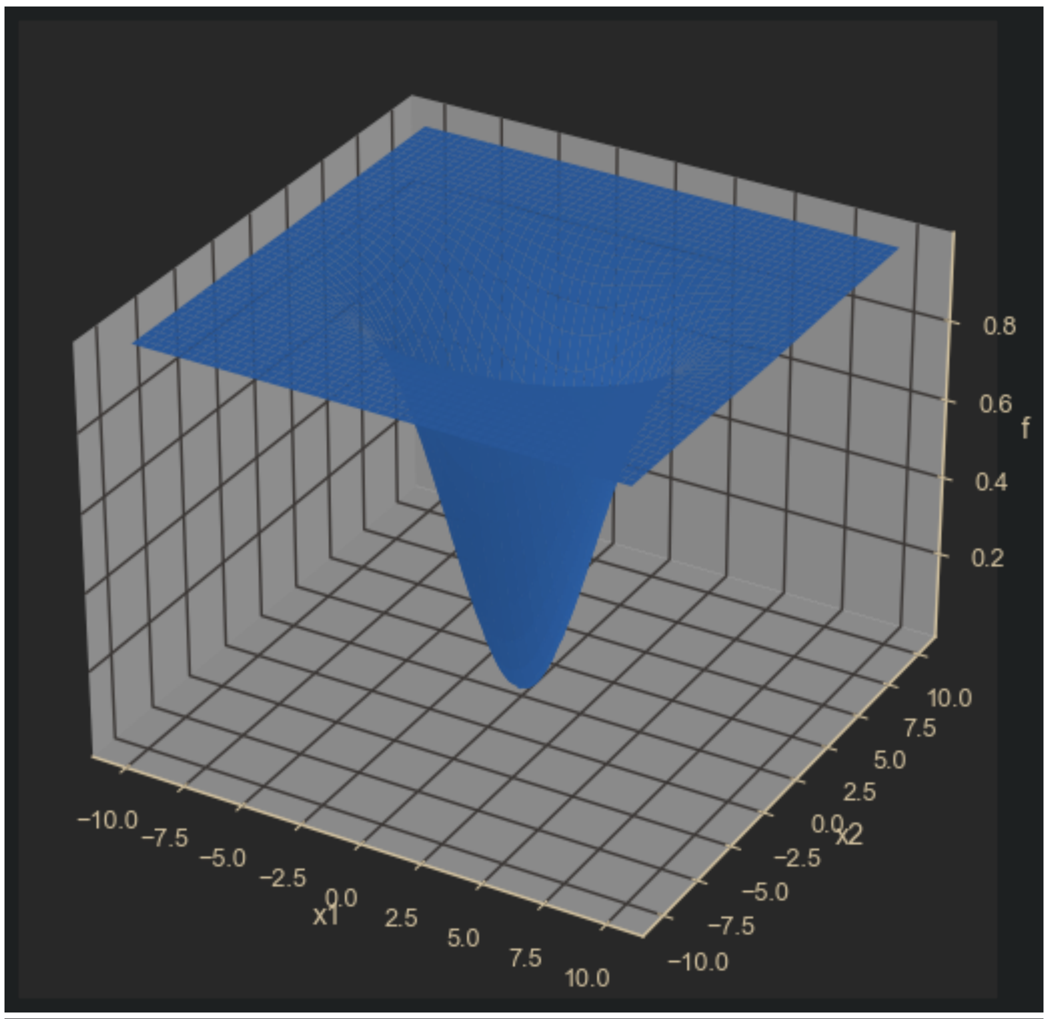
Newton Method

In [3]:

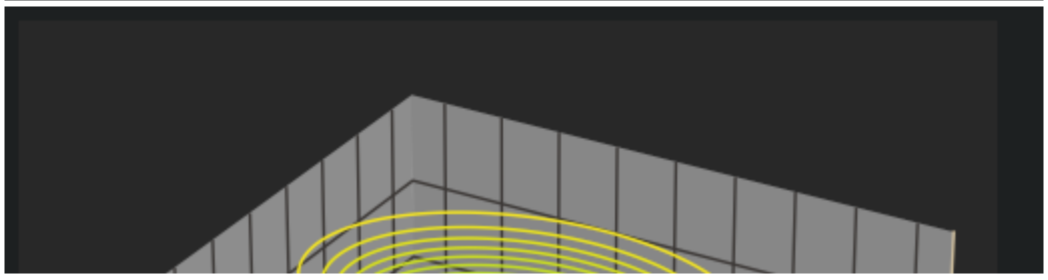
```python
# defining f
def f(x,y,sig_x,sig_y,std_x,std_y):
    return 1-np.exp(-((x-sig_x)**2/(2*std_x)+(y-sig_y)**2/(2*std_y)));
# defining g
def g_x (x,y,sig_x,sig_y,std_x,std_y):
    return f(x,y,sig_x,sig_y,std_x,std_y)*((x-sig_x)/std_x);
def g_y (x,y,sig_x,sig_y,std_x,std_y):
    return f(x,y,sig_x,sig_y,std_x,std_y)*((y-sig_y)/std_y);
# defining H
def H_xx (x,y,sig_x,sig_y,std_x,std_y):
    return f(x,y,sig_x,sig_y,std_x,std_y)*(1/std_x+((x-sig_x)/std_x)**2);
def H_yy (x,y,sig_x,sig_y,std_x,std_y):
    return f(x,y,sig_x,sig_y,std_x,std_y)*(1/std_y+((y-sig_y)/std_y)**2);
def H_xy (x,y,sig_x,sig_y,std_x,std_y):
    return f(x,y,sig_x,sig_y,std_x,std_y)*((y-sig_y)/std_y)*((x-sig_x)/std_x);
# defiing parameters for f (X)
sig_x = 0
sig_y = 0
std_x = 8
std_y = 4
# Generating x and y, 1000 data points from -10 to 10
x = np.linspace(-10,10,1000);
y = np.linspace(-10,10,1000);
# meshgrid for plotting 3D plots
X,Y = np.meshgrid(x,y);
# generating f(x,y)
fxy = f(X,Y,sig_x,sig_y,std_x,std_y);
# plotting Surface plot
plt.figure(figsize=(9,9))
ax = plt.axes(projection='3d')
ax.plot_surface(X, Y, fxy)
ax.set_xlabel('x1');
ax.set_ylabel('x2');
ax.set_zlabel('f');
#plotting contour 3D plot
plt.figure(figsize=(9,9))
ax = plt.axes(projection='3d')
ax.contour3D(X, Y, fxy,80)
ax.set_xlabel('x1');
ax.set_ylabel('x2');
ax.set_zlabel('f');
# array of initial values of x ,y
x_init_arr = [1];
y_init_arr = [1];
# counting iterations
iter = 0;
# looping over inital points
for x_init in x_init_arr:
    for y_init in y_init_arr:
        x_hist = np.array([]); # history array to store all values of x in each update
        y_hist = np.array([]); # history array to store all values of y in each update
        x_hist = np.append(x_hist,x_init); # appending initial value to array
        y_hist = np.append(y_hist,y_init); # appending initial value to array
        # loop to get updates till convergence
        while (g_x(x_hist[-1],y_hist[-1],sig_x,sig_y,std_x,std_y)>= 0.001 or
               g_x(x_hist[-1],y_hist[-1],sig_x,sig_y,std_x,std_y)<=-0.001 or
               g_y(x_hist[-1],y_hist[-1],sig_x,sig_y,std_x,std_y)>= 0.001 or
               g_y(x_hist[-1],y_hist[-1],sig_x,sig_y,std_x,std_y)<=-0.001):
            x_old = x_hist[-1]; # old x before update
```

```
np.array([[H_xx(x_old,y_old,sig_x,sig_y,std_x,std_y),H_xy(x_old,y_old,sig_x,sig_y,std_x,std_y)],

[H_xy(x_old,y_old,sig_x,sig_y,std_x,std_y),H_yy(x_old,y_old,sig_x,sig_y,std_x,std_y)]]);
            dk = -1*(np.linalg.pinv(Hk)@gk)
            alpha_k = 0.1;
            x_up = x_old + alpha_k*dk[0]; # updating x
            y_up = y_old + alpha_k*dk[1]; # updating y
            x_hist = np.append(x_hist,x_up); # appending updated x to history array
            y_hist = np.append(y_hist,y_up); # appending updated y to history array
            iter+=1;
        # defining figure and size
        plt.figure(figsize=(8,6))
        # plotting 2D contour plots
        plt.contour(X, Y, fxy,30);
        # plotting history of updates
        plt.plot(x_hist,y_hist,'r')
        # plotting optimal value
        plt.plot(x_hist[-1],y_hist[-1],color='r',marker='v')
        # plotting initial value
        plt.plot(x_hist[0],y_hist[0],color='r',marker='o')
        # title
        plt.title("x_init= "+str(x_init)+", y_init= "+str(y_init)
                  +"\nf= "+str(f(x_hist[-1],y_hist[-1],sig_x,sig_y,std_x,std_y))+", iter=
"+str(iter))
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_5036/2820456927.py in <module>
     67             x_up = x_old + alpha_k*dk[0]; # updating x
     68             y_up = y_old + alpha_k*dk[1]; # updating y
---> 69             x_hist = np.append(x_hist,x_up); # appending updated x to history array
     70             y_hist = np.append(y_hist,y_up); # appending updated y to history array
     71             iter+=1;

<__array_function__ internals> in append(*args, **kwargs)

~\anaconda3\envs\py38gpu\lib\site-packages\numpy\lib\function_base.py in append(arr, values, axis)
   4743         values = ravel(values)
   4744         axis = arr.ndim-1
-> 4745     return concatenate((arr, values), axis=axis)
   4746 
   4747 

<__array_function__ internals> in concatenate(*args, **kwargs)

KeyboardInterrupt: 
```



0.8
0.6
0.4
0.2
f
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
x1
10.0
7.5
5.0
2.5
0.0
x2
−2.5
−5.0
−7.5
−10.0



In [ ]: