Numerical Exercise 1

First Name: Md Shariar Imroze

Last Name: Khan

Matriculation Number: 220202354

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FormatStrFormatter
```

In [2]:

```
#defining the Rosenbrock function
def ros_func(X):
    x, y = X
    return (1-x)**2 + 100 * (y-x**2)**2

#Gradiant
def ros_func_grad(X):
    x, y = X
    return np.array([
        2*(x-1)-400*x*(y-x**2),
        200*(y-x**2)
    ])

#Hessian
def ros_func_hess(X):
    x, y = X
    return np.matrix([
        [2-400*(y-3*x**2), -400*x],
        [-400*x, 200]
    ])
```

In [3]:

```
#Gradiant descent algorithm
#grad: gradiant function
#x0: starting point
#alpha: learning rate
#tol: termination condition
#max_iter: 1e5
def grad_descent(grad, x0, alpha = 0.001, tol= 1e-5, max_iter=100000):
    x = x0
    for i in range(max_iter):
        x = x - alpha*grad(x)
        #print(x) #check updates
        if np.linalg.norm(grad(x)) < tol:
            return x, i+1
    return np.array(x), max_iter
```

In [4]:

```
#random number generator
#x0 = np.random.default_rng()
#x0= x0.integers(5, size=2)
#x0 = np.zeros(2)

#Starting point
x0 = np.array([2,2])
print('Starting point=',x0)
x_min, iterations = grad_descent(ros_func_grad, x0, alpha = 0.001, max_iter = 100000)
print('x*=',x_min)
print('ros_func(x*)=',ros_func(x_min))
print('ros_func_grad(x*)=',ros_func_grad(x_min))
print('Number of iterations =', iterations)
```

```
Starting point= [2 2]
x*= [1.00001118 1.0000224 ]
ros_func(x*)= 1.251811977306727e-10
ros_func_grad(x*)= [4.46457124e-06 8.94710768e-06]
Number of iterations = 26508
```

According to gradient descent algorithm with the starting point of x0 = [2,2], the algorithm converged after 26508 iteration.

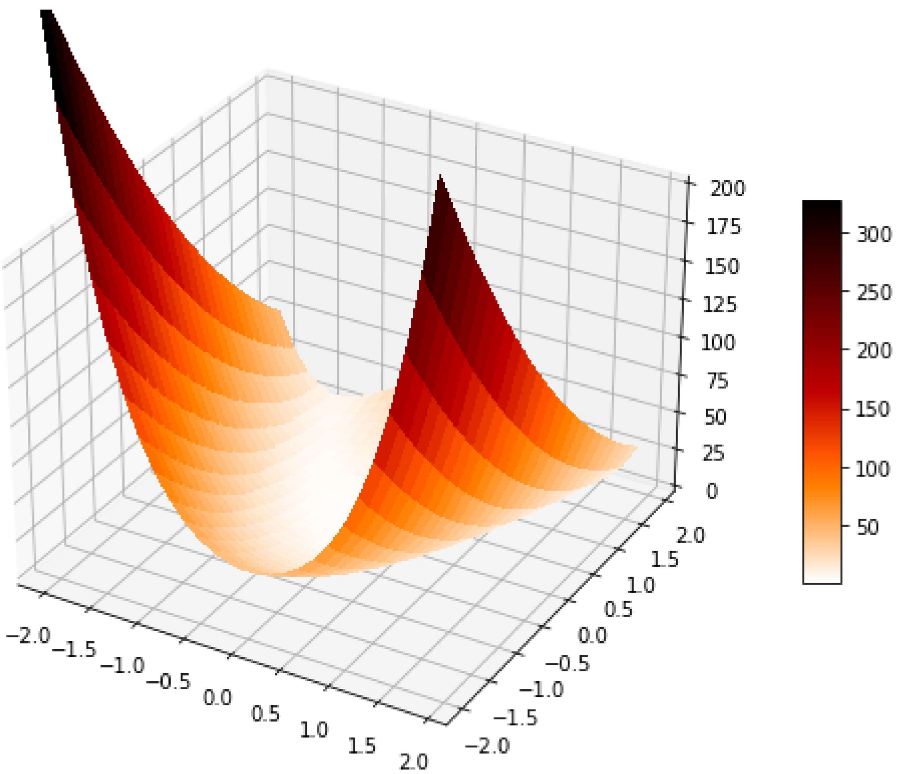
In [5]:

```
# Plotting the Rosenbrock function
ros_func = lambda x,y: (x-1)**2 + 10*(y-x**2)**2;
figRos = plt.figure(figsize=(14, 7))
axRos = figRos.gca(projection='3d')

# Evaluate function
X = np.arange(-2,2, 0.15)
Y = np.arange(-2,2, 0.15)
X, Y = np.meshgrid(X, Y)
Z = ros_func(X,Y)

# Plot the surface
surf = axRos.plot_surface(X, Y, Z, cmap=cm.gist_heat_r,linewidth=0, antialiased=False)
axRos.set_zlim(0, 200)
figRos.colorbar(surf, shrink=0.5, aspect=10)
plt.show()
```

```
C:\Users\shari\AppData\Local\Temp/ipykernel_19964/557123689.py:4: Matplotlib
DeprecationWarning: Calling gca() with keyword arguments was deprecated in M
atplotlib 3.4. Starting two minor releases later, gca() will take no keyword
arguments. The gca() function should only be used to get the current axes, o
r if no axes exist, create new axes with default keyword arguments. To creat
e a new axes with non-default arguments, use plt.axes() or plt.subplot().
  axRos = figRos.gca(projection='3d')
```



In [6]:

```python
#Visualizing the iteration process
def grad_descent(grad, x0, alpha = 0.001, tol= 1e-5, max_iter=100000):
    x = x0
    l = []
    for i in range(max_iter):
        x = x - alpha*grad(x)
        l.append(x)
        #print(x) #check updates
        #saving updates in x1, y1
    return l
```

In [7]:

```python
l = grad_descent(ros_func_grad, x0, alpha = 0.001, max_iter = 27000)
print(l)
```

```
[array([0.398, 2.4  ]), array([0.75606608, 1.9516808 ]), array([1.173916
, 1.67567182]), array([1.31330786, 1.61615322]), array([1.25561837, 1.637
87808]), array([1.28589519, 1.62561797]), array([1.27096845, 1.6311996
6]), array([1.27847879, 1.62803189]), array([1.27461   , 1.62932711]), ar
ray([1.27645524, 1.62838782]), array([1.27541719, 1.62857785]), array([1.
27582998, 1.62820009]), array([1.27551203, 1.6281085 ]), array([1.2755618
, 1.62787298]), array([1.27542655, 1.62770997]), array([1.27538438, 1.627
51055]), array([1.27529542, 1.6273295 ]), array([1.27523   , 1.6271392
9]), array([1.27515277, 1.62695374]), array([1.27508149, 1.62676591]), ar
ray([1.27500724, 1.62657929]), array([1.2749345 , 1.62639212]), array([1.
27486102, 1.62620529]), array([1.27478793, 1.62601836]), array([1.2747146
6, 1.62583154]), array([1.2746415 , 1.62564473]), array([1.27456831, 1.62
545797]), array([1.27449515, 1.62527125]), array([1.27442199, 1.6250845
8]), array([1.27434885, 1.62489794]), array([1.27427572, 1.62471135]), ar
ray([1.2742026, 1.6245248]), array([1.27412949, 1.62433829]), array([1.27
40564 , 1.62415183]), array([1.27398331, 1.6239654 ]), array([1.27391025,
1.62377902]), array([1.27383719, 1.62359268]), array([1.27376414, 1.62340
638]), array([1.27369111, 1.62322012]), array([1.27361809, 1.62303391]),
array([1.27354508, 1.62284773]), array([1.27347209, 1.6226616 ]), array
```

In [8]:

```
# Initialize contour
plt.figure(figsize=(20, 10))
X1 = np.arange(-3,3, 0.15)
Y1 = np.arange(-3,3, 0.15)
X1, Y1 = np.meshgrid(X1, Y1)
Z1 = ros_func(X1,Y1)
plt.contour(X1,Y1,Z1,200)

#Plot initial point x0 in red
plt.plot([x0[0]],[x0[1]],marker='o',markersize=15, color ='r')

#Plotting Descent direction
plt.plot(l[0:10],marker='<',markersize=10, color ='b')
plt.show()
```