EE2703 week6

G.Sai Krishna Reddy <ee21b049>

March 24, 2023

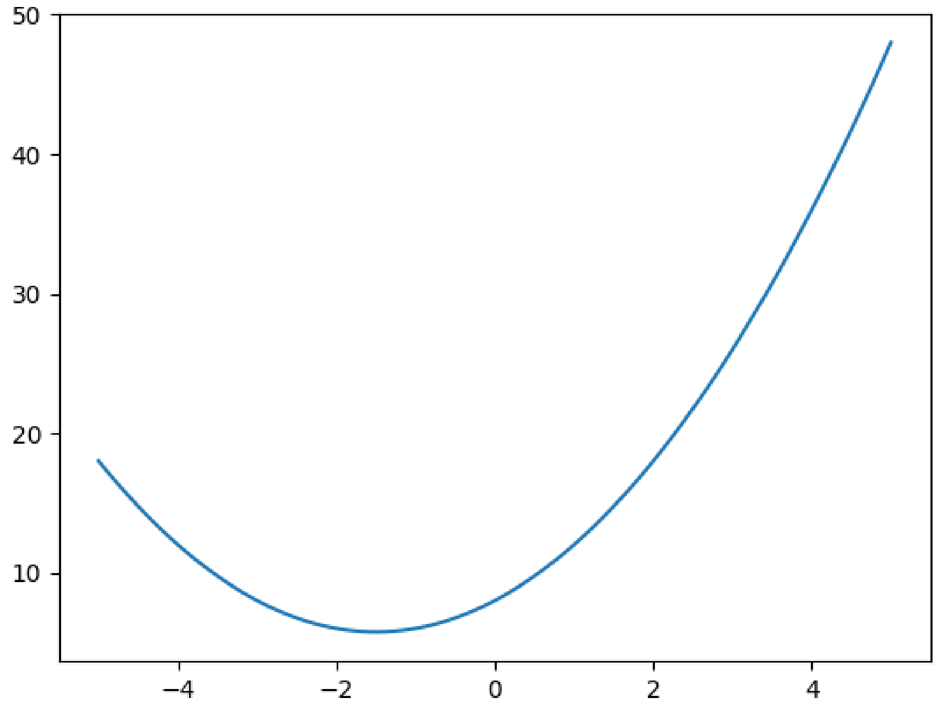
[1]:
```
%matplotlib ipympl
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
from mpl_toolkits.mplot3d import Axes3D
# The following imports are assumed for the rest of the problems
import numpy as np
from numpy import cos, sin, pi, exp
```

0.1 Problem 1 - 1-D simple polynomial

[2]:
```
def f1(x):
    return x ** 2 + 3 * x + 8
def f2(x):
    return 2 * x + 3
xbase = np.linspace(-5, 5, 100)
ybase = f1(xbase)
```

[3]:
```
# Set up some large value for the best cost found so far
b1 = 1000
# Generate several values within a search 'space' and check whether the new␣
↪value is better
# than the best seen so far.
B1 = 4
rangemin, rangemax = -5, 5
fig, ax = plt.subplots()
ax.plot(xbase, ybase)
xall, yall = [], []
lnall,  = ax.plot([], [], 'ro')
lngood, = ax.plot([], [], 'go', markersize=10)

# Learning rate
lr = 0.04

def onestepderiv(frame):
    global b1,B1, lr
    x = B1 - f2(B1) * lr
```

```
    B1 = x
    y = f1(x)
    lngood.set_data(x, y)
    xall.append(x)
    yall.append(y)
    lnall.set_data(xall, yall)
    return lngood,

ani= FuncAnimation(fig, onestepderiv, frames=range(500), interval=400,␣
 ↪repeat=False)
plt.show()
```



[11]:
```
print(f"we get the minimum value at x = {B1} and the minimum value is {f1(B1)}")
```

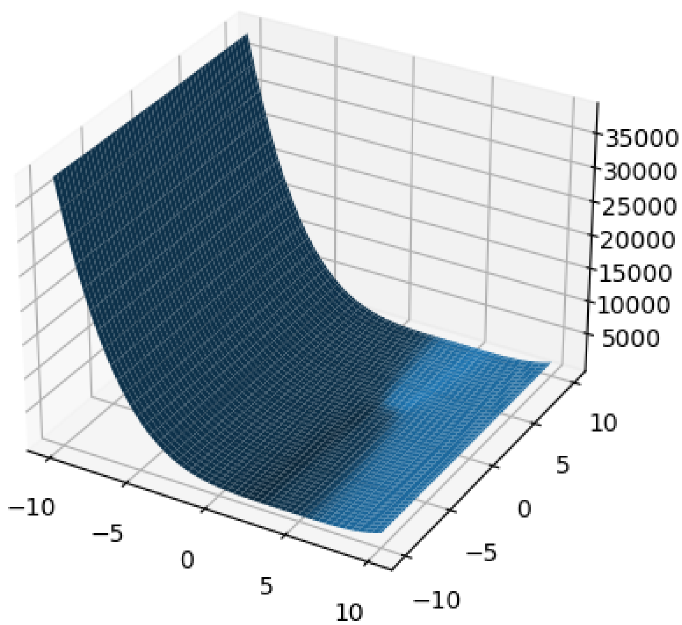
```
we get the minimum value at x = -1.499989556660176 and the minimum value is
5.750000000109063
```

0.2 Problem 2 - 2-D polynomial

[4]:
```python
xlim3 =  [-10, 10]
ylim3 =  [-10, 10]
def f3(x, y):
    return x**4 - 16*x**3 + 96*x**2 - 256*x + y**2 - 4*y + 262

def df3_dx(x, y):
    return 4*x**3 - 48*x**2 + 192*x - 256

def df3_dy(x, y):
    return 2*y - 4
xbase=np.linspace(-10,10,100)
ybase=np.linspace(-10,10,100)
z=f3(xbase,ybase)
```

[10]:
```python
b2 = 100000
Bx,By = -4,0
rangemin, rangemax = -5, 5

# Learning rate
lr = 0.0025
xmesh, ymesh = np.meshgrid(xbase, ybase)
zmesh = f3(xmesh, ymesh)
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(xmesh, ymesh, zmesh)

xall, yall, zall = [], [], []
lnall,  = ax.plot([], [], [], 'ro')
lngood, = ax.plot([], [], [], 'go',markersize=10)

def onestepderiv(frame):
    global b2, Bx, By, lr
    x = Bx - df3_dx(Bx, By) * lr
    y = By - df3_dy(Bx, By) * lr
    Bx, By = x, y
    z = f3(x, y)
    lngood.set_data([x], [y])
    lngood.set_3d_properties([z])
    xall.append(x)
    yall.append(y)
    zall.append(z)
    lnall.set_data(xall, yall)
    lnall.set_3d_properties(zall)
    # return lngood,
```

```
ani = FuncAnimation(fig, onestepderiv, frames=range(1000), interval=10,␣
↪repeat=False)



plt.show()
```



[7]:
```
print(f"we get the minimum value of the funtion at x = {Bx} ,y = {By} and the␣
↪minimum value is z = {f3(Bx,By)}")
```

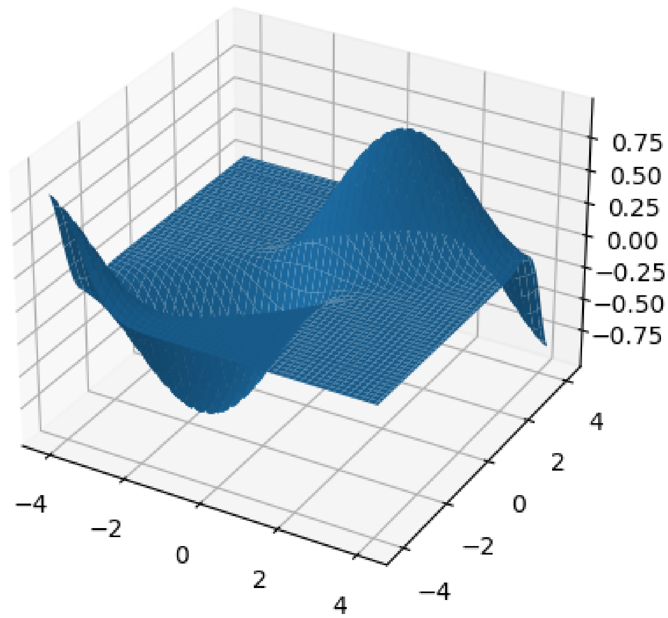
```
we get the minimum value of the funtion at x = 3.7774947518407496 ,y =
1.9867586025281243 and the minimum value is z = 2.0026264346410585
```

0.3 Problem 3 - 2-D function

[11]:
```
xlim4 = [-pi, pi]
def f4(x,y):
    return exp(-(x - y)**2)*sin(y)

def df4_dx(x, y):
    return -2*exp(-(x - y)**2)*sin(y)*(x - y)
```

```python
def df4_dy(x, y):
    return exp(-(x - y)**2)*cos(y) + 2*exp(-(x - y)**2)*sin(y)*(x - y)
```

[12]:
```python
b3 = 100000

xbase = np.linspace(-pi-1, pi+1, 100)
ybase = np.linspace(-pi-1, pi+1, 100)
xmesh, ymesh = np.meshgrid(xbase, ybase)
zmesh = f4(xmesh, ymesh)
Bx, By = -0.1,-0.1
# Learning rate
lr = 0.01

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(xmesh, ymesh, zmesh)

xall, yall, zall = [], [], []
lnall,  = ax.plot([], [], [], 'ro')
lngood, = ax.plot([], [], [], 'go', markersize=10)

def onestepderiv(frame):
    global b3,Bx, By, lr
    x = Bx - df4_dx(Bx,By) * lr
    y = By - df4_dy(Bx, By) * lr
    Bx, By = x, y
    z = f4(x, y)
    lngood.set_data([x], [y])
    lngood.set_3d_properties([z])
    xall.append(x)
    yall.append(y)
    zall.append(z)
    lnall.set_data(xall, yall)
    lnall.set_3d_properties(zall)
    # return lngood,


ani = FuncAnimation(fig, onestepderiv, frames=range(10000), interval=1,␣
 ↪repeat=False)
plt.show()
```




```
[10]: print(f"we get the minimum value of the funtion at x = {Bx} ,y = {By} and the␣
      ↪minimum value is z = {f4(Bx,By)}")
```

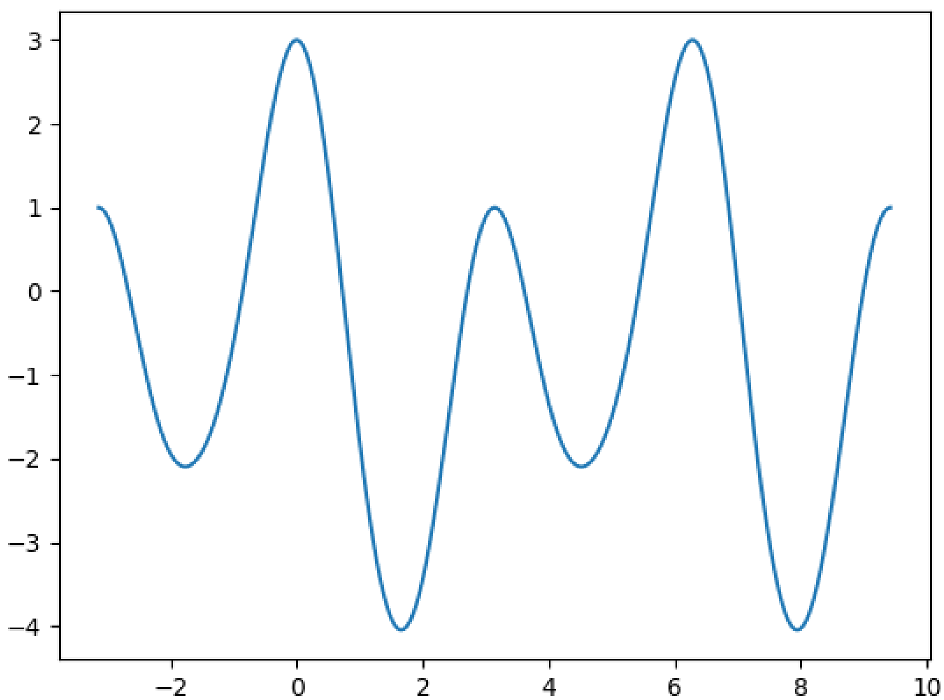
```
we get the minimum value of the funtion at x = -1.570796326794869 ,y =
-1.5707963267948746 and the minimum value is z = -1.0
```

0.4 Problem 4 - 1-D trigonometric

```
[16]: def f5(x):
          return cos(x)**4 - sin(x)**3 - 4*sin(x)**2 + cos(x) + 1
      def df5_dx(x):
          return -1*sin(x)*(4*cos(x)**3+3*sin(x)*cos(x)+8*cos(x)+1)
```

```
[17]: b4 = 100000

      Bx = 2.9
      rangemin, rangemax = -5, 5
      xbase = np.linspace(-pi, 3*pi, 1000)
      ybase=f5(xbase)
      fig, ax = plt.subplots()
      ax.plot(xbase, ybase)
```

```
xall, yall = [], []
lnall,  = ax.plot([], [], 'ro')
lngood, = ax.plot([], [], 'go', markersize=10)

# Learning rate
lr = 0.04


def onestepderiv(frame):
    global b4, Bx, lr
    x = Bx - df5_dx(Bx) * lr
    Bx = x
    y = f5(x)
    lngood.set_data(x, y)
    xall.append(x)
    yall.append(y)
    lnall.set_data(xall, yall)
    # return lngood,

ani= FuncAnimation(fig, onestepderiv, frames=range(20), interval=500,␣
 ↪repeat=False)
plt.show()
```



[12]:
```
print(f"we get the minimum value at x = {Bx} and the minimum value is {f5(Bx)}")
```

```
we get the minimum value at x = 1.661702611487221 and the minimum value is
-4.0454120419887385
```

0.5 Gradient Descent

[ ]:
```
def gradient_descent(range, f, df, lr=0.1, n=100):

    range = np.array(range)
    for i in range(n):
        gradient = df(*range)
        range =range - lr * gradient
    return range.tolist(), f(*range)
optimal_values, minimum_value = gradient_descent(range, f, df)

# Print the results
print(f"The optimal values are: {optimal_values}")
print(f"The minimum value found is: {minimum_value}")
```

Here the range parameter is expected to be a tuple or list of initial values for the variables of the function.And number of elements in range array is equal to number of variables. In each iteration it calculates different values of variables using learning rate(lr) and moves towards minimum value of the function.

The same is implemented in all the above problems.