23915-MDSC-103P-ESE

Qu1)

a)

Formulation of the Problem:

Objective	x1	x2	z		
Coefficient	7	10			
Solution	360	300	5520		
c1	5	6	3600	<=	3600
c2	1	2	960	<=	960
c3	1	0	360	<=	500
c4	0	1	300	<=	500

After solving the linear programming problem, we got the optimal solution z = 5520 when x1 = 360 and x2 = 300.

Sensitivity Report:

Variable Cells

Cell	Name	Final Value	Reduced Cost	Objective Coefficient	Allowable Increase	Allowable Decrease
B3	Solution x1	360	0	7	1.333333333	2
C3	Solution x2	300	0	10	4	1.6

Constraints

Cell	Name	Final Value	Shadow Price	Constraint R.H. Side	Allowable Increase	Allowable Decrease
D5	c1 z	3600	1	3600	280	720
D6	c2 z	960	2	960	160	93.33333333
D7	c3 z	360	0	500	1E+30	140
D8	c4 z	300	0	500	1E+30	200

b) Cost Coefficient Sensitivity Analysis

The objective function is z = 7x1 + 10x2.

From the variable Cells table, we can see that the allowable increase in the coefficient of x1 is 1.33333333333333 and the allowable decrease is 2. Therefore, the optimality range of x1 such that the optimal solution doesn't change is (7-2, 7+1.33333333333333) = (5, 8.33333333333333). This means that if the price of a dozen baseballs (x1) ranges from 5 to 8.33333333333333 dollars, this implies that the optimal solution x1 = 360 and x2 = 300 will remain the same.

The allowable increase in the coefficient of x2 is 4 and the allowable decrease is 1.6. Therefore, the optimality range of x2 such that the optimal solution doesn't change is (10-1.6, 10+4) = (8.4, 14). We

can infer that if the price of a dozen softballs (x2) ranges from 8.4 to 14 dollars, this implies that the optimal solution will remain the same.

c) Right Hand Side Sensitivity Analysis

From the Constraints table, we can see that the shadow price of only constraint c1 and c2 are given, which are 1 and 2 respectively. Shadow price is also known as unit worth of resource which means that how much revenue is being incurred from a unit resource.

Therefore, for each square foot of cowhide sheets, a revenue of 1 dollar is being incurred. Likewise, for every minute of production, a revenue of 2 dollars is incurred. The shadow price of constraint c1 and c2 is 0 which means that there's no revenue made from a unit of resource.

The feasibility range of constraint c1 is (3600-720, 3600+280) = (2880, 3880). Within this range, the optimal solution will change but the profit made will remain the same. The feasibility range of constraint c2 is (960- 93.3333333333333, 960+160) = (866.6666666666666, 1120). The feasibility range of constraint c3 is (360, infinity). The feasibility range of constraint c4 is (300, infinity). Within the specific feasibility ranges of each constraint, the profit made will not change.

Qu2)

```python
import numpy as np
import matplotlib.pyplot as plt
```

0.5s Python

```python
# Let x1 = x and x2 = y
f = lambda x, y: 4*x + 6*y - 2*((x)**2) - 2*x*y - 2*((y)**2)
```

0.0s Python

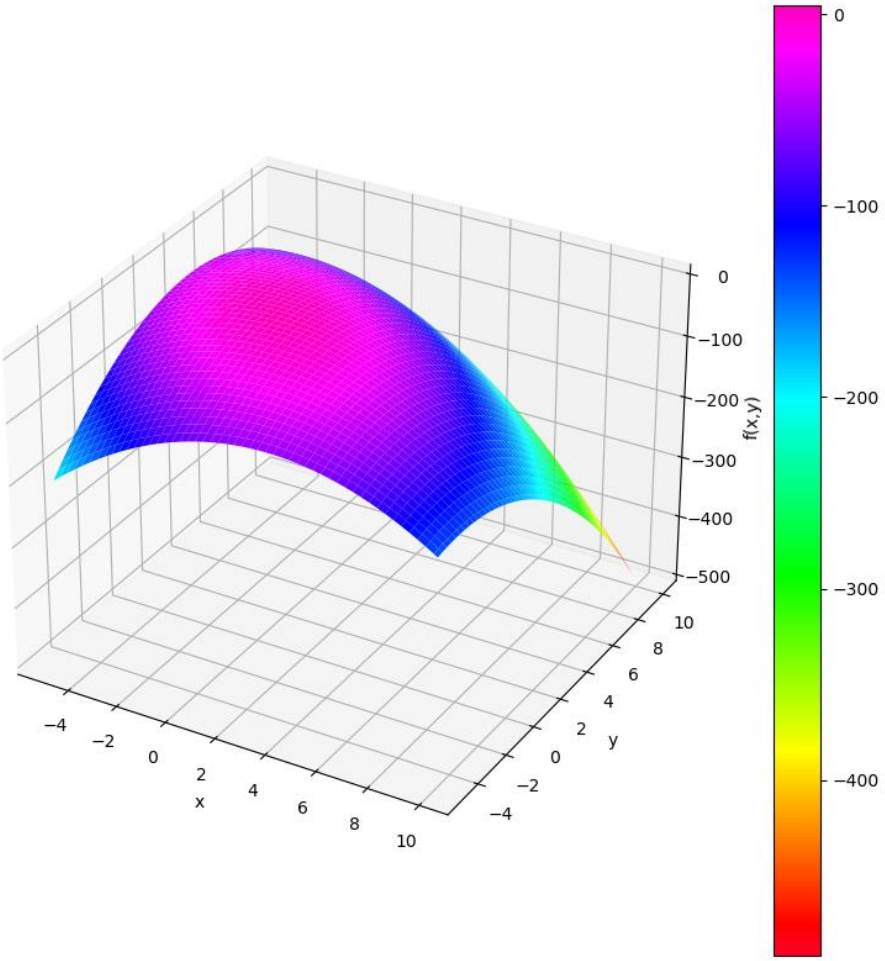
```python
x = np.linspace(-5,10,100)  # range of values of x
y = np.linspace(-5,10,100)  # range of values of y

X,Y = np.meshgrid(x,y)      #creating a meshgrid
F = f(X,Y)                  #saving the function values with respect to corresponding values of x and y

figure = plt.figure(figsize=(10,10))    #specifying the size of the figure
ax = plt.subplot(projection='3d')   # specifying that the graph should be 3-dimensional
graph = ax.plot_surface(X,Y,F, cmap= 'gist_rainbow')    # plotting the graph
figure.colorbar(graph)
ax.set_xlabel("x")
ax.set_ylabel("y")
ax.set_zlabel("f(x,y)")
plt.show()

```

0.6s Python




```python
# Using Gradient Ascent to maximize the function
delta = 1 * (10**(-5))
learning_rate = 0.1
deriv_x = 4 - 4*x - 2*y     # partial derivative of function f with respect to x
deriv_y = 6 - 2*x - 4*y     # partial derivative of function f with respect to y
X0 = np.array([1,1])    # initial point

while (i == 0):
    X1 = X0 + learning_rate * np.array([4 - 4*X0[0] - 2* X0[1], 6 - 2*X0[0] - 4*X0[1]])
    if(np.abs(X1[0] - X0[0]) < delta and np.abs(X1[1] - X0[1]) < delta):    # if the difference between X0 and X1 is less than delta,
        i = 1                                                               # we come out of the function
X0 = X1

print(X1)

# The function is maximum at x = 0.8 and y = 1
```

✓ 0.0s Python

```
[0.8 1. ]
```