MDSC-103

Final Lab Test

❖ Formulate the problem in the Excel file and generate the sensitivity analysis

Let **x1 = no. of dozens of baseballs, x2 = no. of dozens of softballs**

Max: $Z = 7x1 + 10x2$

Subj to:

$5x1 + 6x2 <= 3600$

$X1 + 2x1 <= 960$

$0 <= x1 <= 500, 0 <= x2 <= 500$

				no.of dozens of baseballs	no.of dozens of softballs			
				x1	x2			
				360	300			
Max		Z	:	7	10		=	5520
Subj to:		c1	:	5	6	3600	<=	3600
		c2	:	1	2	960	<=	960
		c3	:	-	-	360	<=	500
		c4	:	-	-	300	<=	500

The sensitivity report for the above simplex problem is:

Variable Cells

Cell	Name	Final Value	Reduced Cost	Objective Coefficient	Allowable Increase	Allowable Decrease
H7	x1	360	0	7	1.333333333	2
I7	x2	300	0	10	4	1.6

Constraints

Cell	Name	Final Value	Shadow Price	Constraint R.H. Side	Allowable Increase	Allowable Decrease
J11	:	3600	1	3600	280	720
J12	:	960	2	960	160	93.33333333
J13	-	360	0	500	1E+30	140
J14	-	300	0	500	1E+30	200

- Write on cost coefficient sensitivity analysis.
  - For x1:
    - The price of baseball can be increased by a maximum of 1.33 units until which the solution remains optimal.
    - Can be decreased by a maximum 2 units until which the solution remains optimal.
    - Any increment or decrement beyond the above values will change the optimal solution.
  - For x2:
    - The price of softball can be increased by a maximum of 4 units until which the solution remains optimal.
    - Can be decreased by a maximum of 1.6 units until which the solution remains optimal
    - Any increment or decrement beyond the above values will change the optimal solution.

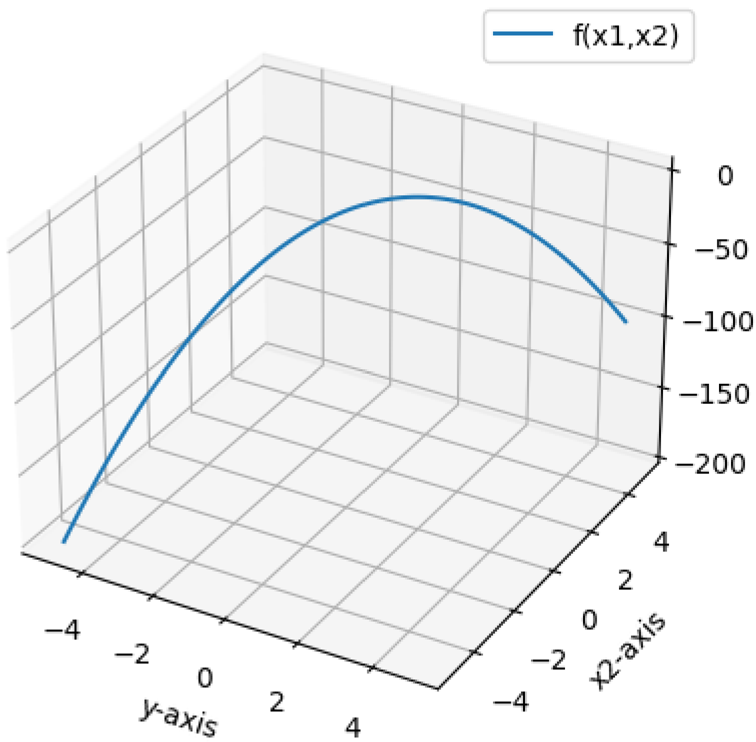
The reduced cost for both the balls is 0 which is means there is no additional profit associated with their production.

- Write on Right Hand Side Sensitivity Analysis
  - For c1(cowhide):
    - Can be increased its usage by a maximum amount of “280” square feet and for each increase it contributes 1 unit towards objective function since its shadow price is 1.
    - Can be decreased by a maximum of “720” below.
  - For c2(time):
    - An additional amount of “160” minutes can be used and for each additional minute it contributes 2 units towards the objective function.
    - Can be decreased by a maximum of “93.3” minutes.

C4 and C3 can be increased by any amount but does not have any effect in maximizing the profit since their shadowed price is 0.

C3 and C4 can be decreased by a maximum of 140 and 200 respectively without any effect on the profit, below which the optimal solution changes.

```
import numpy as np
import scipy as sc
import matplotlib.pyplot as plt

# defining the mathematical function using lambda func
f = lambda x1,x2: 4*x1 + 6*x2 -2*(x1**2) - 2*x1*x2 - 2*(x2**2)

# creating a sub plot for 3d visualization
ax = plt.figure().add_subplot(projection='3d')

# generating a sample data points for x1 in the range (-5,5)
x1 = np.linspace(-5,5,100)

# generating a sample data points for x2 in the range (-5,5)
x2 = np.linspace(-5,5,100)

# computing f(x1,x2) for all the points of x1 and x2 generated above using function f
y = f(x1,x2)

# plotting the curve in 3d
ax.plot(x1, x2, y, label='f(x1,x2)')
ax.legend()

# setting the axis labels
ax.set_xlabel('x1-axis')
ax.set_ylabel('x2-axis')
ax.set_xlabel('y-axis')
plt.show()
```



```python
# initial approximation
x0 = np.array([0,0])

# learning rate
eta = 0.1
iter = 0

# finding new x point for each iteration using the learning rate of 0.1
xn = lambda x: x - eta*x

# the function is taken as -f since minimizing -f means maximizing f
f = lambda x1,x2: -4*x1 - 6*x2 + 2*(x1**2) + 2*x1*x2 + 2*(x2**2)

# the partial derivatives of f wrt x1 and x2
df_x1 = lambda x1,x2: -4 + 4*x1 + 2*x2
df_x2 = lambda x1,x2: -6 + 2*x1 + 4*x2

while iter < 100:

    # calculating the gradient in both x1 and x2 direction
    x = np.array([df_x1(x0[0],x0[1]),df_x2(x0[0],x0[1])])
    temp = list(x)

    # if the gradient vanishes in both the directions we break since that would be the maximum point as it is gradient ascent
    if temp == [0.0,0.0]:
        break
    # otherwise find the new x0 , does until gradient vanishes
    x0 = xn(x)

    iter = iter + 1

print('maximum occured at the point ',x,' and the value is ',f(x[0],x[1]))
```

```
maximum occured at the point  [-23957794.63079219 -23958684.74759885]  and the value is  3443983734009102.0
```