Umm Al-Qura University Computers
College Computer Science & Artificial
Intelligence Department

BSc Artificial Intelligence
Optimization and Constraint
Programming
1445 - 2nd Trimester

Simple Optimization Problem Formulation & Solution

Click on the link

GROUP 4

A real-life example of an optimization problem in the healthcare field, specifically in pharmaceutical manufacturing.

we identifying the maximum profit in the manufacturing company.

```
## 1.1 Problem Description
A company manufactures two types of medicines, X1 and X2. Each medicine
requires 3 materials, A, B and C. X1 requires 5g of material A and 15g of
material B and 8g of material C, while X2 requires 10g of material A and
25g of material B and 12g of material C.
The available quantity of materials is 50g of material A and 80g of
material B and 60 of material C.
To prevent the materials from spoiling, the company needs to manufacture 6
of medicines x1 and x2 a day.
The manufacturing price of X1 is USD 110, and the price of X2 is USD 85,
What is the **max protit** of the two medications??
```

1.1 Problem Description

A company manufactures two types of medicines, X1 and X2. Each medicine requires 3 materials, A, B and C. X1 requires 5g of material A and 15g of material B and 8g of material C, while X2 requires 10g of material A and 25g of material B and 12g of material C. The available quantity of materials is 50g of material A and 80g of material B and 60 of material C. To prevent the materials from spoiling, the company needs to manufacture 6 of medicines x1 and x2 a day. The manufacturing price of X1 is USD 110, and the price of X2 is USD 85, What is the **max protit** of the two medications??

```
## 1.2 Optimization Problem Formulation
There will be Two variables:
*   $x_1$ : Number of medicine a to be produced
*   $x_2$ : Number of medicine b to be produced

Objective function:

* profit= $110 x_1 + 85 x_2$


Constraints:
*   $ 5 x_1 + 15 x_2 \le 50$
*   $ 10 x_1 + 25 x_2 \le 80$
*   $ 8 x_1 + 12 x_2 \le 60$
*   $ x_1 + x_2 = 6$
*   $ x_1  \ge 0$
*   $ x_2  \ge 0$





Objective:maximize the profit subject to the given constraints

Solve the given linear programming problem:
$$
\begin{align}
\max_{x_1,x_2} \quad & P = 110 x_1 + 85 x_2 \\
\textrm{s.t.}  \quad & 5 x_1 + 15 x_2 \le 50 \\
 & 10 x_1 + 25 x_2 \le 80 \\
 & 8 x_1 + 12 x_2 \le 60\\
 & x_1 + x_2 = 6\\
 & x_1 \le 0\\
 & x_2 \le 0
\end{align}
$$
```

1.2 Optimization Problem Formulation

There will be Two variables:

- x_1 : Number of medicine a to be produced
- x_2 : Number of medicine b to be produced

Objective function:

- profit= $110x_1 + 85x_2$

Constraints:

1 - $5x_1 + 15x_2 \le 50$
2 - $10x_1 + 25x_2 \le 80$
3 - $8x_1 + 12x_2 \le 60$
4 - $x_1 + x_2 = 6$
5 - $x_1 \ge 0$
6 - $x_2 \ge 0$

Objective:maximize the profit subject to the given constraints

Solve the given linear programming problem:

$$
\begin{align}
\max_{x_1,x_2} \quad & P = 110 x_1 + 85 x_2 \\
\textrm{s.t.} \quad & 5 x_1 + 15 x_2 \le 50 \\
 & 10 x_1 + 25 x_2 \le 80 \\
 & 8 x_1 + 12 x_2 \le 60 \\
 & x_1 + x_2 = 6 \\
 & x_1 \le 0 \\
 & x_2 \le 0
\end{align}
$$

Here, we have defined the number of variables, objective function, and constraints.

1/2/3-The quantity of material A/B/C required to manufacture drug x1 is multiplied by the number of produced drugs, plus the quantity of material A/B/C required to manufacture drug x2 multiplied by the number of produced drugs.

4-To prevent material spoilage, the company manufactures exactly 6 drugs daily.

5/6-The quantity of manufactured drugs cannot be negative.

Here, we already have the mathematical formula for the objective function. Note that we can only show two variables: X1 and X2 We will assume that X2 is on the y-axis. To be able to plot the constraints.

we have to re-arrange the given constraints into the form:
X2 >= X1.:

```
## 1.3 Graphical Solution using Corner Point Method

$$
\begin{align}
x_2 & \le \frac{50 - 5x_1}{15} & = 3.33 - 0.33x_1 \\
x_2 & \le \frac{80 - 10 x_1}{25}   & = 3.2 - 0.4x_1 \\
x_2 & \le \frac{60 - 8 x_1}{12}   & = 5 - 0.66x_1 \\
x_2 & = 6 - x_1\\
\end{align}
$$
```

1.3 Graphical Solution using Corner Point Method

$$
\begin{align}
x_2 & \le \frac{50 - 5x_1}{15} & = 3.33 - 0.33x_1 \\
x_2 & \le \frac{80 - 10 x_1}{25} & = 3.2 - 0.4x_1 \\
x_2 & \le \frac{60 - 8 x_1}{12} & = 5 - 0.66x_1 \\
x_2 & = 6 - x_1
\end{align}
$$

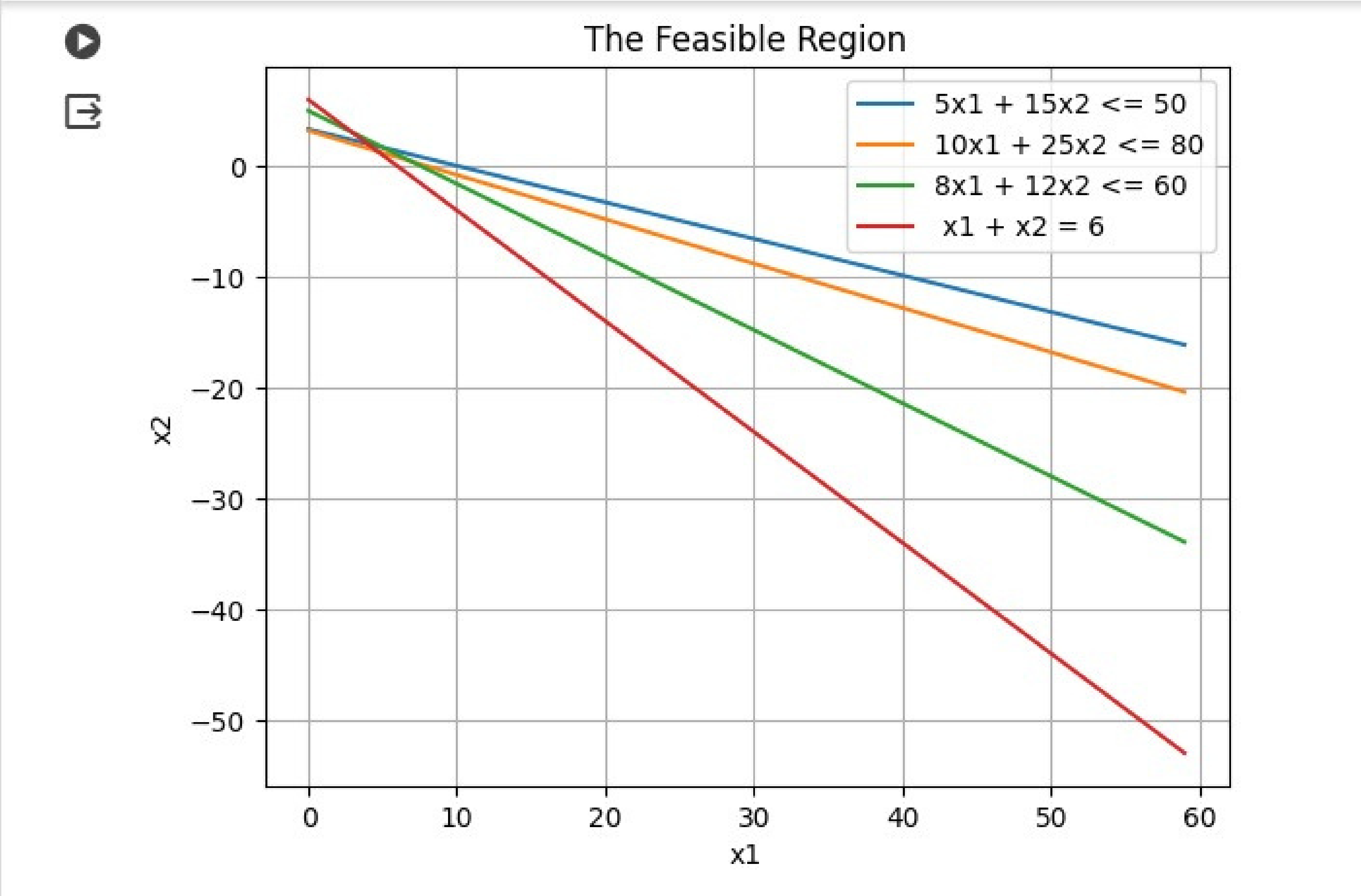
```python
## import the required libraries
import matplotlib.pyplot as plt
import numpy as np
```

```python
#### Define some points for x1
x = np.arange(0,60)
print("x1 values to be used for plotting:",x)

#### (optional) Get a handle for your figure and axis
fig, ax = plt.subplots()

#### plot function simply gets two arrays: x-axis and y-axis coordinates.
#### We also add a label for each constraint for reference

#constraint for material A
ax.plot(x, 3.33 - 0.33 * x, label = '5x1 + 15x2 <= 50')
#constraint for material B
ax.plot(x, 3.2 - 0.4 * x, label= '10x1 + 25x2 <= 80')
#constraint for material C
ax.plot(x, 5 - 0.66 * x, label= '8x1 + 12x2 <= 60')
#constraint for the number of medicines x1 and x2 the company manufacture a day
ax.plot(x, 6 - x, label= ' x1 + x2 = 6')

####settings for the graph:
#### show the grid and legend
##ax.set_xlim([5, 0])
##ax.set_ylim([6, 4])
ax.set(title="The Feasible Region", xlabel="x1", ylabel="x2")
ax.grid(True)
ax.legend()
#### show the figure
plt.show()
```

```
x1 values to be used for plotting: [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
 48 49 50 51 52 53 54 55 56 57 58 59]
```

Assuming numbers from zero to sixty for variable X.

we then plotted the constraints for materials A,B and C for the X1 and X2 medicines.



And this is the figure that appeared to us after plotting the constraints.

```
## install ipyml
!pip install ipympl

## import the handle for the output channel
from google.colab import output

## enable third-party widget on jupyter
output.enable_custom_widget_manager()
```

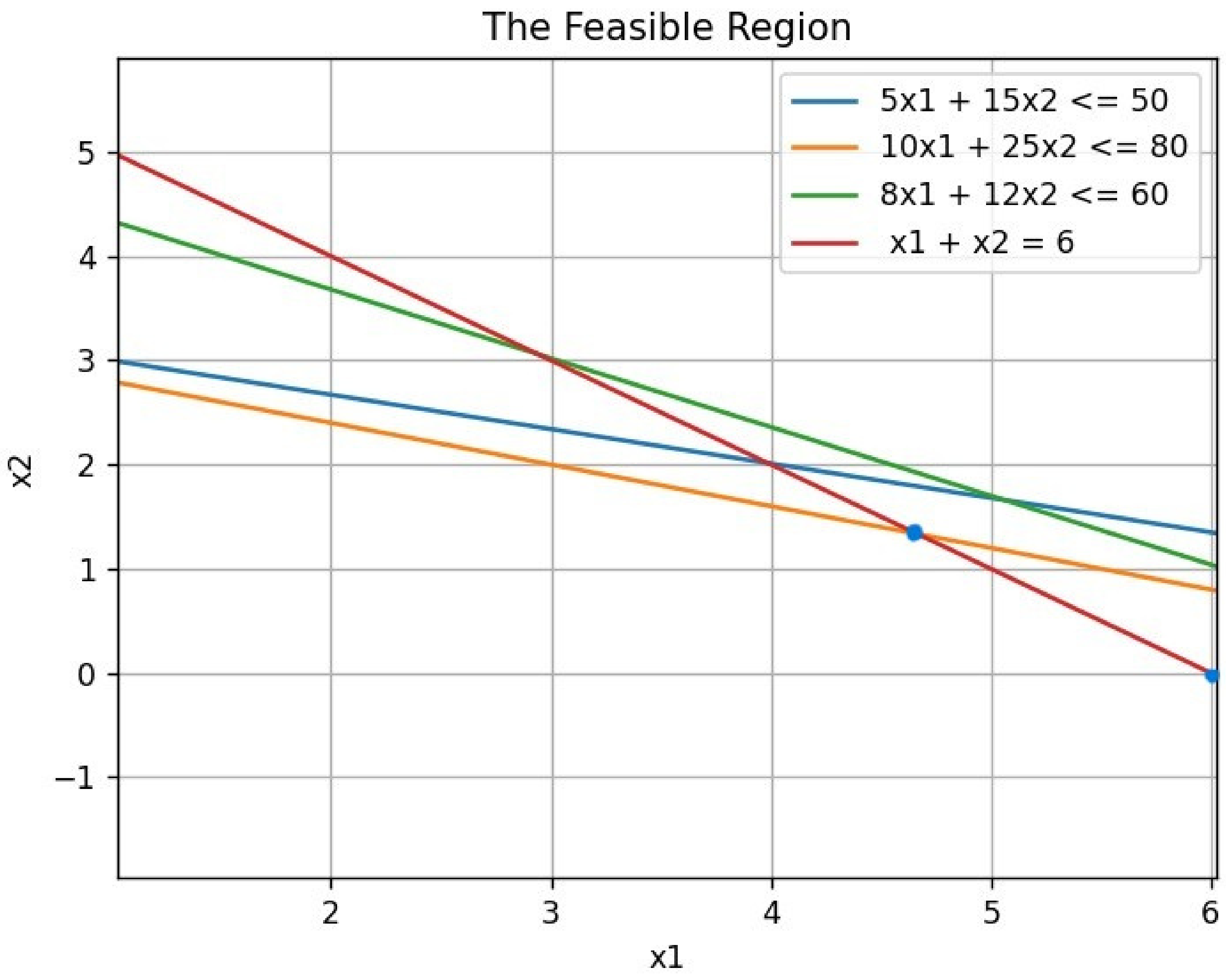
```
Collecting ipympl
  Downloading ipympl-0.9.3-py2.py3-none-any.whl (511 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 511.6/511.6 kB 5.8 MB/s eta 0:00:00
Requirement already satisfied: ipython<9 in /usr/local/lib/python3.10/dist-packages (from ipympl) (7.34.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from ipympl) (1.23.5)
Requirement already satisfied: ipython-genutils in /usr/local/lib/python3.10/dist-packages (from ipympl) (0.2.0)
Requirement already satisfied: pillow in /usr/local/lib/python3.10/dist-packages (from ipympl) (9.4.0)
Requirement already satisfied: traitlets<6 in /usr/local/lib/python3.10/dist-packages (from ipympl) (5.7.1)
Requirement already satisfied: ipywidgets<9,>=7.6.0 in /usr/local/lib/python3.10/dist-packages (from ipympl) (7.7.1)
Requirement already satisfied: matplotlib<4,>=3.4.0 in /usr/local/lib/python3.10/dist-packages (from ipympl) (3.7.1)
Requirement already satisfied: setuptools>=18.5 in /usr/local/lib/python3.10/dist-packages (from ipython<9->ipympl) (67.7.2)
Collecting jedi>=0.16 (from ipython<9->ipympl)
  Downloading jedi-0.19.1-py2.py3-none-any.whl (1.6 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.6/1.6 MB 12.6 MB/s eta 0:00:00
Requirement already satisfied: decorator in /usr/local/lib/python3.10/dist-packages (from ipython<9->ipympl) (4.4.2)
Requirement already satisfied: pickleshare in /usr/local/lib/python3.10/dist-packages (from ipython<9->ipympl) (0.7.5)
Requirement already satisfied: prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from ipython<9->ipympl) (3.0.43)
Requirement already satisfied: pygments in /usr/local/lib/python3.10/dist-packages (from ipython<9->ipympl) (2.16.1)
Requirement already satisfied: backcall in /usr/local/lib/python3.10/dist-packages (from ipython<9->ipympl) (0.2.0)
Requirement already satisfied: matplotlib-inline in /usr/local/lib/python3.10/dist-packages (from ipython<9->ipympl) (0.1.6)
Requirement already satisfied: pexpect>4.3 in /usr/local/lib/python3.10/dist-packages (from ipython<9->ipympl) (4.9.0)
Requirement already satisfied: ipykernel>=4.5.1 in /usr/local/lib/python3.10/dist-packages (from ipywidgets<9,>=7.6.0->ipympl) (5.5.6)
Requirement already satisfied: widgetsnbextension~=3.6.0 in /usr/local/lib/python3.10/dist-packages (from ipywidgets<9,>=7.6.0->ipympl) (3.6.6)
Requirement already satisfied: jupyterlab-widgets>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from ipywidgets<9,>=7.6.0->ipympl) (3.0.9)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib<4,>=3.4.0->ipympl) (1.2.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib<4,>=3.4.0->ipympl) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib<4,>=3.4.0->ipympl) (4.46.0)
```

Some settings so we can determine the Corners accurately

```
[4] ## activate the ipymlp backend
    %matplotlib ipympl
```

```
## get the figure and axis handeles
fig2, ax2 = plt.subplots()

## define a callback function to be called once the user clicked on the figure
def onclick(event):
    ix, iy = event.xdata, event.ydata
    print("x1=", ix, "x2=", iy)

## connect the callback function to the event manager
cid = fig2.canvas.mpl_connect('button_press_event', onclick)

## plot the constraints and fix your graph settings as before
ax2.plot(x, 3.33 - 0.33* x, label = '5x1 + 15x2 <= 50')
ax2.plot(x, 3.2 - 0.4 * x, label= '10x1 + 25x2 <= 80')
ax2.plot(x, 5 - 0.66 * x, label= '8x1 + 12x2 <= 60')
ax2.plot(x, 6 - x, label= ' x1 + x2 = 6')
####settings for the graph:
####show the grid and legend
##ax2.set_xlim([10, 0])
##ax2.set_ylim([5, 0])
ax2.set(title="The Feasible Region", xlabel="x1", ylabel="x2")
ax2.grid(True)
ax2.legend()

plt.show()
```

We ploted the constraints again to determine the corners that help us find the maximum profit



These are the corners we chose because they are in the feasible region

```
[6] ## get coordinates of the feasible region into a seperate arrays: x1 values and x2 values
    corners_x1 = np.array([6.00, 4.64])
    corners_x2 = np.array([0.00, 1.35])

    ## evaluate the objective function for all corner points in a single vector computation
    P = 110 * corners_x1 + 85 * corners_x2
    print(P)

    [660.   625.15]
```

```
[7] ## print the values for x1 and x2 that gives the maximum Z value.
    ####  Z.argmax returns the index where Z has the maximum value.
    max_index = P.argmax()
    print("solutions: x1=" , corners_x1[max_index], ", x2=", corners_x2[max_index])

    print("solutions (rounded): x1= %.0f, x2= %.0f" % (corners_x1[max_index], corners_x2[max_index]))

    solutions: x1= 6.0 , x2= 0.0
    solutions (rounded): x1= 6, x2= 0
```

We have set the coordinates of the selected corners in an array of x1 and array of x2
Then we multiplied the x1 array by the price of the medicine x1 and sum it with the array x2 multiplied by its price and save

the result array in variable P
in variable max_index we but the the maximum value of P array
Then we printed the results

1.4 Using PuLP (`pulp`)

```
!pip install pulp
```

```
Collecting pulp
  Downloading PuLP-2.7.0-py3-none-any.whl (14.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 14.3/14.3 MB 40.4 MB/s eta 0:00:00
Installing collected packages: pulp
Successfully installed pulp-2.7.0
```

[9]
```
from pulp import *
```

[10]
```
# Create the model
model = LpProblem(name="EX1-problem", sense=LpMaximize)

# Initialize the decision variables
x1 = LpVariable(name="x1", lowBound=0)
x2 = LpVariable(name="x2", lowBound=0)

# We added the objective function and the constraints to the model
model += (110 * x1 + 85* x2)        ### objective function
model += (5 * x1 + 15 * x2 <= 50)    ### constraint 1
model += (10 * x1 + 25 * x2 <= 80)   ### constraint 2
model += (8 * x1 + 12 * x2 <= 60)    ### constraint 3
model += (x1 + x2 == 6)              ### constraint 4
```

Here we imported the pulp library, then we set the name and the lower bound of x1 and x2. Then we put in the constraints

```
[11] ## check the model content
     status = model.solve()
```

```
[12] print('status:', LpStatus[model.status])
     print('objective:', model.objective.value())

     for var in model.variables():
       print(var.name, ":", var.value())
```

```
status: Optimal
objective: 660.0
x1 : 6.0
x2 : 0.0
```

```
model.solver.toDict()
```

```
{'solver': 'PULP_CBC_CMD',
 'mip': True,
 'msg': True,
 'keepFiles': False,
 'warmStart': False,
 'timeMode': 'elapsed'}
```

Here we used a method to solve the problem and then printed the results

```
## 2.1 Problem Description

A company manufactures 10 types of medicines, X1, X2, X3, X4, X5, X6, X7,
X8, X9, X10. Each medicine requires 2 materials, A ,B and C.
X1 requires 5g of material A ,15g of material B and 7g of material C
X2 requires 4g of material A, 6g of material B  and 14g of material C
X3 requires 7g of material A, 9g of material B  and 16g of material C
X4 requires 12g of material A, 13g of material B  and 26g of material C
X5 requires 2g of material A, 8g of material B and 18g of material C
X6 requires 10g of material A, 19g of material B and 22g of material C
X7 requires 17g of material A, 16g of material B and 23g of material C
X8 requires 6g of material A, 3g of material B and 20g of material C
X9 requires 8g of material A, 11g of material B and 21g of material C
X10 requires 3g of material A, 18g of material B and 19g of material C.The
available quantity of materials is 90g of material A, 110g of material B
and 170g of material C.To prevent the materials from spoiling, the company
needs to manufacture 60 of X1, X2, X3, X4, X5, X6, X7, X8, X9 and X10.The
manufacturing profit of X1 is USD 20, X2 is USD 40, X3 is USD 30, X4 is
USD 10, X5 is USD 70, X6 is USD 60, X7 is USD 50, X8 is USD 80, X9 is USD
100, X10 is USD 110.What is the **maximize protit** of the Ten
medications??
```


2.1 Problem Description

A company manufactures 10 types of medicines, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10. Each medicine requires 2 materials, A ,B and C. X1 requires 5g of material A ,15g of material B and 7g of material C X2 requires 4g of material A, 6g of material B and 14g of material C X3 requires 7g of material A, 9g of material B and 16g of material C X4 requires 12g of material A, 13g of material B and 26g of material C X5 requires 2g of material A, 8g of material B and 18g of material C X6 requires 10g of material A, 19g of material B and 22g of material C X7 requires 17g of material A, 16g of material B and 23g of material C X8 requires 6g of material A, 3g of material B and 20g of material C X9 requires 8g of material A, 11g of material B and 21g of material C X10 requires 3g of material A, 18g of material B and 19g of material C.The available quantity of materials is 90g of material A, 110g of material B and 170g of material C.To prevent the materials from spoiling, the company needs to manufacture 60 of X1, X2, X3, X4, X5, X6, X7, X8, X9 and X10.The manufacturing profit of X1 is USD 20, X2 is USD 40, X3 is USD 30, X4 is USD 10, X5 is USD 70, X6 is USD 60, X7 is USD 50, X8 is USD 80, X9 is USD 100, X10 is USD 110.What is the **maximize protit** of the Ten medications??

In the second problem, we extended the first problem by changing the two types of medicine x1 and x2 to 10 types of medicine x1, x2, x3, x4, x5, x6, x7, x8, x9, x10

then we set the constraints according to the company's requirements.

```
## 2.2 Optimization Problem Formulation
There will be 10 variables:
*   $x_1$ : Number of medicine a to be produced
*   $x_2$ : Number of medicine b to be produced
*   $x_3$ : Number of medicine c to be produced
*   $x_4$ : Number of medicine d to be produced
*   $x_5$ : Number of medicine e to be produced
*   $x_6$ : Number of medicine f to be produced
*   $x_7$ : Number of medicine g to be produced
*   $x_8$ : Number of medicine h to be produced
*   $x_9$ : Number of medicine i to be produced
*   $x_10$ : Number of medicine j to be produced

Objective function:
*   Profit = $ 20 x_1 + 40 x_2 + 30 x_3 + 10 x_4 + 70 x_5 + 60 x_6 + 50 x_7 + 80 x_8  +  100 x_9 + 110 x1_0 $



Constraints:
*   $  5x_1 +4x_2 +7x_3 +12x_4 +2x_5 +10x_6 +17x_7 +6x_8  +8x_9 +3x_10 \le 90 $
*   $ 15x_1 +6x_2 +9x_3 +13x_4 +8x_5 +19x_6 +16x_7 +3x_8  +11x_9 +18x_10 \le 110 $
*   $ 7x_1 +14x_2 +16x_3 +26x_4 +18x_5 +22x_6 +23x_7 +20x_8  +21x_9 +19x_10 \le 170 $
*   $ x_1 +x_2 +x_3 +x_4+ x_5 +x_6 +x_7 +x_8 +x_9 +x_10 = 60$
*   $ x_1 +x_2 +x_3 +x_4+ x_5 +x_6 +x_7 +x_8 +x_9 +x_10 \ge 0$


Objective: maximize the profit subject to the given constraints


Solve the given linear programming problem:
$$
\begin{align}
\max_{x_1,x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9 x1_0} \quad & Profit =  20 x_1 + 40 x_2 + 30 x_3 + 10 x_4 + 70 x_5 + 60 x_6 + 50 x_7 + 80 x_8  +  100 x_9 + 110 x_10 \\
\textrm{s.t.}  \quad & 5x_1 +4x_2 +7x_3 +12x_4 +2x_5 +10x_6 +17x_7 +6x_8  +8x_9 +3x_10 \le 90  \\
 & 15x_1 +6x_2 +9x_3 +13x_4 +8x_5 +19x_6 +16x_7 +3x_8  +11x_9 +18x_10 \le 110 \\
  & 7x_1 +14x_2 +16x_3 +26x_4 +18x_5 +22x_6 +23x_7 +20x_8  +21x_9 +19x_10 \le 170 \\
 &x_1 +x_2 +x_3 +x_4+ x_5 +x_6 +x_7 +x_8 +x_9 +x_10 = 60 \\
 & x_1 +x_2 +x_3 +x_4+ x_5 +x_6 +x_7 +x_8 +x_9 +x_10 \ge 0
\end{align}
$$
```

2.2 Optimization Problem Formulation

There will be 10 variables:

- x_1 : Number of medicine a to be produced
- x_2 : Number of medicine b to be produced
- x_3 : Number of medicine c to be produced
- x_4 : Number of medicine d to be produced
- x_5 : Number of medicine e to be produced
- x_6 : Number of medicine f to be produced
- x_7 : Number of medicine g to be produced
- x_8 : Number of medicine h to be produced
- x_9 : Number of medicine i to be produced
- x_10 : Number of medicine j to be produced

Objective function:

- Profit = $20x_1 + 40x_2 + 30x_3 + 10x_4 + 70x_5 + 60x_6 + 50x_7 + 80x_8 + 100x_9 + 110x1_0$

Constraints:

- $5x_1 + 4x_2 + 7x_3 + 12x_4 + 2x_5 + 10x_6 + 17x_7 + 6x_8 + 8x_9 + 3x_10 \le 90$
- $15x_1 + 6x_2 + 9x_3 + 13x_4 + 8x_5 + 19x_6 + 16x_7 + 3x_8 + 11x_9 + 18x_10 \le 110$
- $7x_1 + 14x_2 + 16x_3 + 26x_4 + 18x_5 + 22x_6 + 23x_7 + 20x_8 + 21x_9 + 19x_10 \le 170$
- $x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_10 = 60$
- $x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_10 \ge 0$

Objective: maximize the profit subject to the given constraints

Solve the given linear programming problem:

$$
\begin{aligned}
\max_{x_1,x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9 x1_0} \quad & Profit = 20x_1 + 40x_2 + 30x_3 + 10x_4 + 70x_5 + 60x_6 + 50x_7 + 80x_8 + 100x_9 + 110x_10 \\
\textrm{s.t.} \quad & 5x_1 + 4x_2 + 7x_3 + 12x_4 + 2x_5 + 10x_6 + 17x_7 + 6x_8 + 8x_9 + 3x_10 \le 90 \\
& 15x_1 + 6x_2 + 9x_3 + 13x_4 + 8x_5 + 19x_6 + 16x_7 + 3x_8 + 11x_9 + 18x_10 \le 110 \\
& 7x_1 + 14x_2 + 16x_3 + 26x_4 + 18x_5 + 22x_6 + 23x_7 + 20x_8 + 21x_9 + 19x_10 \le 170 \\
& x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_10 = 60 \\
& x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_10 \ge 0
\end{aligned}
$$

What is required is to extract the products that give us the greatest profit, to do that we multiplyed the number of each drug produced by its price and the result is the profit

1/2/3- The quantity of material A/B/C required to manufacture drug x1, x2..... X10 is multiplied by the number of produced drugs, sumed together

4- To prevent material spoilage, the company manufactures exactly 60 drugs daily.

5/6...15- The quantity of manufactured drugs cannot be negative.

2.3 Using PuLP (`pulp`)

```
[14] !pip install pulp
```

```
Requirement already satisfied: pulp in /usr/local/lib/python3.10/dist-packages (2.7.0)
```

```
[15] from pulp import *
```

```
# Create the model
model = LpProblem(name="EX2-problem", sense=LpMaximize)

# Initialize the decision variables
x1 = LpVariable(name="x1", lowBound=0)
x2 = LpVariable(name="x2", lowBound=0)
x3 = LpVariable(name="x3", lowBound=0)
x4 = LpVariable(name="x4", lowBound=0)
x5 = LpVariable(name="x5", lowBound=0)
x6 = LpVariable(name="x6", lowBound=0)
x7 = LpVariable(name="x7", lowBound=0)
x8 = LpVariable(name="x8", lowBound=0)
x9 = LpVariable(name="x9", lowBound=0)
x10 = LpVariable(name="x10", lowBound=0)

#  We added the objective function and the constraints to the model
model += (20 *x1 +  40 * x2 + 30 *x3 + 10 *x4 + 70 * x5 + 60 * x6 + 50 * x7 + 80  *x8  +  100 *x9 + 110 *x10)  ### objective function
model += ( 5 * x1 +  4 * x2 +  7 * x3 + 12  *x4 + 2  * x5 + 10 *x6 + 17 * x7 + 6 *x8  + 8* x9 + 3* x10 <= 90)  #constraints 1
model += ( 15 * x1 +  6 * x2 +  9 * x3 + 13  *x4 + 8  * x5 + 19  *x6 + 16  * x7 + 3 * x8  + 11 * x9 + 18  *x10 <= 110 ) #constraints 2
model += ( 7 * x1 +  14 * x2 +  16 * x3 + 26  *x4 + 18  * x5 + 22  *x6 + 23  * x7 + 20 * x8  + 21 * x9 + 19  *x10 <= 170 ) #constraints 3
model += ( x1 +   x2 +   x3 +  x4 +   x5 +  x6 +  x7 + x8  + x9 + x10 <= 30) #constraints 4
```

we put the constraints

```
[17] ## check the model content
     status = model.solve()
```

```
[18] print('status:', LpStatus[model.status])
     print('objective:', model.objective.value())

     for var in model.variables():
       print(var.name, ":", var.value())
```

```
status: Optimal
objective: 869.636966
x1 : 0.0
x10 : 5.5775578
x2 : 0.0
x3 : 0.0
x4 : 0.0
x5 : 0.0
x6 : 0.0
x7 : 0.0
x8 : 3.2013201
x9 : 0.0
```

```
model.solver.toDict()
```

```
{'solver': 'PULP_CBC_CMD',
 'mip': True,
 'msg': True,
 'keepFiles': False,
 'warmStart': False,
 'timeMode': 'elapsed'}
```

Then we solved the problem and printed the result.

Difference between
solving two-variable problem **VS** multi-variable problem

COMPARISON	TWO VARIABLES	TEN VARIABLES
Complexity	simple and easy to control 2 variables using basic algebraic or graphical methods	More detailed and a bit harder to control 10 variables
Constraints	Less constraints	More constraints
Type of solution	can be plotted and solve graphically or using a library in python	can't be plot and solve graphically only we can use a library in python
Solution Space	have a solution that can be represented graphically as a point or a line in a two-dimensional	have a solution space that can be represented in three or more dimensions

Difference between

The solution from Python **VS** The graphical solution

COMPARISON	THE SOLUTION FROM PYTHON	BOTH	GRAPHICAL SOLUTION
result	-	both of them give the same result	-
complexity	Requires objective functions and constraint	-	It requires assuming numbers for the input and using the constraints to extract the output, then plotting out the constructs and locate the corners.
difficulty	We need to put the constants as they are.	-	we need to solve the constant equation with respect to one variable so that we can use them to extract the output

We learned how to use the Python language to help us solve complex problems

We learned how to classify the important products to a company according to the profits they return

conclusion

We learned how to write down a problem in a way that makes it easier for us to analyze and solve

conclusion

After this exercise, we can work on realistic problems in real life

We learned to analyze the results of the problems we solved to solve the problem

We learned how to plot the constraints of a problem and from where we can locate the corners

We learned to use linear algebra to solve real-world problems

Group members & ID

444002659	فرح عبد الله الحازمي
444006628	فاطمة أحمد الزهراني
444015444	حليمة عبد الله محمد