Laboratorio 8: MÉTODO SIMPLEX. FORMA ESTÁNDAR. PASOS DEL MÉTODO SIMPLEX

Integrantes:

- Cervera Vasquez, Eslin Yair
- Escriba Flores, Daniel Agustin

In [1]:
```
## Importamos las librerias necesarias
import numpy as np
from scipy.optimize import linprog

# ================================
import warnings
warnings.filterwarnings('ignore')
```

PROBLEMATICA

Prime S. A. es una empresa de la ciudad de Lima que se dedica a la fabricación de carrocerías para automóviles, camiones y furgonetas, la cual consta de dos tipos. En el tipo A, para fabricar la carrocería de un camión, se invierten 4 días/operario, para fabricar la de un automóvil se invierten 2 días/operario y para fabricar una de furgoneta 2 días/operario. En el tipo B se invierten 5 días/operario tanto en carrocerías para camión, automóvil y furgoneta. Por limitaciones de maquinaria en los talleres y mano de obra, el tipo A dispone de 315 días/operario y el tipo B de 290 días/operario. Los beneficios que se obtienen por cada carrocería son: para automóvil `$6125`, para camión `$5550` y para furgoneta `$6125`.

Parte A

Escriba el problema de programación lineal que maximice los beneficios y luego conviértalo a su forma estándar.

Programacion lineal

$x1$: Número de carrocerías de automóviles fabricadas.
$x2$: Número de carrocerías de camiones fabricadas.
$x3$: Número de carrocerías de furgonetas fabricadas.

Maximizar $z = 6125x1 + 5550x2 + 6125x3$

sujeto a

$2x1 + 4x2 + 2x3 <= 315$
$5x1 + 5x2 + 5x3 <= 290$

Forma estandar

$x1$: Número de carrocerías de automóviles fabricadas.
$x2$: Número de carrocerías de camiones fabricadas.
$x3$: Número de carrocerías de furgonetas fabricadas.
$s1$: variable de holgura 1
$s2$: variable de holgura 2

Maximizar $z - 6125x1 - 5550x2 - 6125x3 = 0$

sujeto a

$2x1 + 4x2 + 2x3 + s1 = 315$
$5x1 + 5x2 + 5x3 + s2 = 290$

Parte B

Resuelva el problema de programación lineal haciendo uso de las funciones vistas en clase, pero indicando en la matriz actualizada, los cambios de la fila pivote a columna pivote y que se refleje en la solución final.

In [2]:
```
z = np.array([-6125,-5550,-6125,0,0])
A = np.array([[2,4,2,1,0],
              [5,5,5,0,1]])
b = np.array([315,290,0])


# Etiquetamos las filas y columnas

variables_básicas = ['s1', 's2','z']
columnas = ['x1', 'x2', 'x3', 's1', 's2']
```

In [3]:
```
# Modificamos la funcion tabla, para mostrar los cambios

def tabla_simplex(z, A, b):

    tabla = np.hstack((np.vstack((A,z)),np.vstack(b)))  # Agregar b como última col
    print("\nTabla Simplex:")
    print()
    print(f"{'Var Básica':>9}", end="   ")
```

```
    for var in columnas:
        print(f"{var:>9}", end="   ")
    print(f"{'CR':>9}")
    print()
    for i, fila in enumerate(tabla):
        print(f"{variables_básicas[i]:>9}", end="   ")
        for valor in fila:
            print(f"{valor:>9.3f}", end="   ")
        print() #
        print() # Doble espaciado
```

In [4]:
```
def encontrar_columna_pivote(z):
    val = np.min(z)
    col = np.argmin(z)
    return val, col
```

In [5]:
```
def encontrar_fila_pivote(A, b, col):
    filas, columnas = np.shape(A)
    ratio = []

    for i in range(filas):
        ratio_val = b[i]/A[i,col]
        if ratio_val > 0:
            ratio.append(ratio_val)

    return ratio.index(min(ratio))
```

In [6]:
```
def actualizar_tabla(z, A, b, fil, col):
    filas, columnas = np.shape(A)
    A = A.astype(float)

    # Obtener un 1 en la intersección de la fila y columna pivote
    pivote = A[fil,col]
    A[fil] = A[fil]/pivote
    b[fil] = b[fil]/pivote

    # Actualizar los valores de las ecuaciones
    for i in range(filas):
        if A[i,col] != 0 and i != fil:
            b[i] = b[i] - (b[fil]*A[i,col])
            A[i] = A[i] - (A[fil]*A[i,col])

    z = z - (A[fil]*z[col])

    return z, A, b
```

In [7]:
```
def actualizar_tabla(z, A, b, fil, col):
    filas, columnas = np.shape(A)
    A = A.astype(float)

    # Obtener un 1 en la intersección de la fila y columna pivote
    pivote = A[fil,col]
    A[fil] = A[fil]/pivote
    b[fil] = b[fil]/pivote
```

```
    # Actualizar los valores de las ecuaciones
    for i in range(filas):
        if A[i,col] != 0 and i != fil:
            b[i] = b[i] - (b[fil]*A[i,col])
            A[i] = A[i] - (A[fil]*A[i,col])

    # Actualizar la función objetivo
    b[filas] = b[filas] - (b[fil]*z[col])
    z = z - (A[fil]*z[col])

    return z, A, b
```

In [8]:

```
# Agregamos funcion decoracion para ordenar
def Linea_de_separacion():
    return print("\n"+"======="*10+"\n")

# Mostrar la tabla inicial
print("Tabla inicial (Simplex):")
tabla_simplex(z, A, b)
print(f"Filas de Restriciones: {variables_básicas}")
Linea_de_separacion()
val, col = encontrar_columna_pivote(z)
iteracion = 1


# Bucle Simplex
while val < 0:

    print(f"\nIteración {iteracion}:")
    print()
    # Encontrar la fila pivote
    fil = encontrar_fila_pivote(A, b, col)
    print(f"Columna pivote: {col+1}, Fila pivote: {fil+1} (Variable entrante: {colu

    # Mostrar qué variable entra y qué variable sale
    var_saliente = variables_básicas[fil]
    var_entrante = columnas[col]
    print(f"Variable {var_saliente} sale de la base y la variable {var_entrante} en
    print()
    print()

    # Actualizar las variables básicas y no básicas
    variables_básicas[fil] = var_entrante

    # Actualizar la tabla con el pivote
    z, A, b = actualizar_tabla(z, A, b, fil, col)

    # Mostrar la tabla actualizada
    tabla_simplex(z, A, b)
    print(f"Filas de Restriciones actualizadas: {variables_básicas}")
    Linea_de_separacion()

    # Encontrar la nueva columna pivote
    val, col = encontrar_columna_pivote(z)
    iteracion += 1
```

```python
# Mostrar la tabla final y solución
print("\nTabla final (solución óptima):")
tabla_simplex(z, A, b)


# Calcular los valores finales de las variables

variables = ['x1', 'x2', 'x3','z']

solucion = np.zeros(4)  # Solo para las variables x1, x2, x3
for i in range(len(variables_básicas)):
    if variables_básicas[i] in variables:  # Solo para x1, x2, x3
        indice = variables.index(variables_básicas[i])
        solucion[indice] = b[i]

# Mostrar la solución final
print("\nSolución final:")
for i, valor in enumerate(solucion):
    print(f"{variables[i]} = {valor}")
```

```
Tabla inicial (Simplex):

Tabla Simplex:

Var Básica          x1          x2          x3          s1          s2          CR

       s1       2.000       4.000       2.000       1.000       0.000     315.000

       s2       5.000       5.000       5.000       0.000       1.000     290.000

        z   -6125.000   -5550.000   -6125.000       0.000       0.000       0.000

Filas de Restriciones: ['s1', 's2', 'z']

========================================================================


Iteración 1:

Columna pivote: 1, Fila pivote: 2 (Variable entrante: x1)
Variable s2 sale de la base y la variable x1 entra en la base.



Tabla Simplex:

Var Básica          x1          x2          x3          s1          s2          CR

       s1       0.000       2.000       0.000       1.000      -0.400     199.000

       x1       1.000       1.000       1.000       0.000       0.200      58.000

        z       0.000     575.000       0.000       0.000    1225.000   355250.000

Filas de Restriciones actualizadas: ['s1', 'x1', 'z']

========================================================================


Tabla final (solución óptima):

Tabla Simplex:

Var Básica          x1          x2          x3          s1          s2          CR

       s1       0.000       2.000       0.000       1.000      -0.400     199.000

       x1       1.000       1.000       1.000       0.000       0.200      58.000

        z       0.000     575.000       0.000       0.000    1225.000   355250.000


Solución final:
x1 = 58.0
x2 = 0.0
```

```
x3 = 0.0
z = 355250.0
```

Parte C

Verifique que se obtienen los mismos resultados por medio de la función 'linprog' de la librería 'scipy'.

In [9]:
```
from scipy.optimize import linprog
z = np.array([-6125, -5550,-6123])
A = np.array([[2, 4, 2], [5,5,5]])
b = np.array([315,290])
```

In [10]:
```
resul = linprog(z,A,b,method="revised simplex")
print(resul)
```

```
message: Optimization terminated successfully.
success: True
 status: 0
    fun: -355250.0
      x: [ 5.800e+01  0.000e+00  0.000e+00]
    nit: 1
```

In [11]:
```
    print('z_max = {0:.1f}, x1 = {1:.1f}, x2 = {2:.1f}, x3 = {3:.1f}'.
      format(-resul.fun,resul.x[0],resul.x[1],resul.x[2]))
```

```
z_max = 355250.0, x1 = 58.0, x2 = 0.0, x3 = 0.0
```

Conclusion:

Nuestro análisis comparativo revela que la implementación manual del método Simplex y el método 'revised simplex' producen resultados coincidentes, lo que confirma la corrección y eficiencia de nuestra implementación