Autor: Renan Tonolli Mondini - RA: 191010324

Sistemas Lineares

Importando as bibliotecas:

In [1]:
```
import numpy as np
from sympy import *
x1, x2, x3, x4, x5 = symbols('x1, x2, x3, x4, x5')
```

Definindo a matriz triangular superior:

In [2]:
```
linha_1 = [1, 1, 1, 0, 2]
linha_2 = [0, 1, 2, 1, 1]
linha_3 = [0, 0, 1, 2, 1]
linha_4 = [0, 0, 0, 2, 1]
linha_5 = [0, 0, 0, 0, 10]

matriz = Matrix([linha_1, linha_2, linha_3, linha_4, linha_5])
matriz
```

Out[2]:

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 2 \\ 0 & 1 & 2 & 1 & 1 \\ 0 & 0 & 1 & 2 & 1 \\ 0 & 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 0 & 10 \end{bmatrix}$$

In [3]:
```
linha_x = Matrix([x1,x2,x3,x4,x5])
linha_x
```

Out[3]:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}$$

In [4]:
```
linha_r = Matrix([1,2,-2,-4,-20])
linha_r
```

Out[4]:

$$\begin{bmatrix} 1 \\ 2 \\ -2 \\ -4 \\ -20 \end{bmatrix}$$

Resolvendo a matriz triangular superior:

In [5]:
```
solve(Eq(matriz * linha_x , linha_r), [x1, x2, x3, x4, x5])
```

Out[5]: `{x1: 2, x2: 1, x3: 2, x4: -1, x5: -2}`

Portanto, $x_1 = 2$, $x_2 = 1$, $x_3 = 2$, $x_4 = -1$, $x_5 = -2$.

Definindo a matriz triangular inferior:

In [6]:
```
linha_1 = [1, 0, 0, 0, 0]
linha_2 = [1, 1, 0, 0, 0]
linha_3 = [1, 2, 1, 0, 0]
linha_4 = [1, 1, 2, 2, 0]
linha_5 = [-2, -1, 0, -3, 10]

matriz = Matrix([linha_1, linha_2, linha_3, linha_4, linha_5])
matriz
```

Out[6]:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 2 & 1 & 0 & 0 \\ 1 & 1 & 2 & 2 & 0 \\ -2 & -1 & 0 & -3 & 10 \end{bmatrix}$$

In [7]:

```
linha_x = Matrix([x1,x2,x3,x4,x5])
linha_x
```

Out[7]:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}$$

In [8]:

```
linha_r = Matrix([[1],[3],[6],[9],[0]])
linha_r
```

Out[8]:

$$\begin{bmatrix} 1 \\ 3 \\ 6 \\ 9 \\ 0 \end{bmatrix}$$

Resolvendo a matriz triangular inferior:

In [9]:

```
solve(Eq(matriz * linha_x , linha_r), [x1, x2, x3, x4, x5])
```

Out[9]:

```
{x1: 1, x2: 2, x3: 1, x4: 2, x5: 1}
```

Portanto, $x_1 = 1$, $x_2 = 2$, $x_3 = 1$, $x_4 = 2$, $x_5 = 1$.

Método de Gauss

Transformando o Sistema Linear em Matrizes:

In [10]:

```
linha_1 = [2, 1, 3, 4]
linha_2 = [1, 4, 2, 1]
linha_3 = [3, 2, 1, 4]
linha_4 = [2, 2, 3, 1]

matriz = Matrix([linha_1, linha_2, linha_3, linha_4])
matriz
```

Out[10]:

$$\begin{bmatrix} 2 & 1 & 3 & 4 \\ 1 & 4 & 2 & 1 \\ 3 & 2 & 1 & 4 \\ 2 & 2 & 3 & 1 \end{bmatrix}$$

In [11]:

```
linha_x = Matrix([[x1],[x2],[x3],[x4]])
linha_x
```

Out[11]:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

In [12]:

```
linha_r = Matrix([[16],[5],[11],[12]])
linha_r
```

Out[12]: $\begin{bmatrix} 16 \\ 5 \\ 11 \\ 12 \end{bmatrix}$

Calculando o Determinante da Matriz $A_1 = \begin{vmatrix} 2 \end{vmatrix}$

In [13]:
```
linha_1 = [2]

matriz_A1 = Matrix([linha_1])
matriz_A1.det()
```

Out[13]: 2

Determinante de $\begin{vmatrix} 2 \end{vmatrix} = 2$

Calculando o Determinante da Matriz $A_2 = \begin{vmatrix} 2 & 1 \\ 1 & 4 \end{vmatrix}$

In [14]:
```
linha_1 = [2, 1]
linha_2 = [1, 4]

matriz_A2 = Matrix([linha_1, linha_2])
matriz_A2.det()
```

Out[14]: 7

Determinante de $\begin{vmatrix} 2 & 1 \\ 1 & 4 \end{vmatrix} = 7$

Calculando o Determinante da Matriz $A_3 = \begin{vmatrix} 2 & 1 & 3 \\ 1 & 4 & 2 \\ 3 & 2 & 1 \end{vmatrix}$

In [15]:
```
linha_1 = [2, 1, 3]
linha_2 = [1, 4, 2]
linha_3 = [3, 2, 1]

matriz_A3 = Matrix([linha_1, linha_2, linha_3])
matriz_A3.det()
```

Out[15]: -25

Determinante de $\begin{vmatrix} 2 & 1 & 3 \\ 1 & 4 & 2 \\ 3 & 2 & 1 \end{vmatrix} = -25$

Como os determinantes são diferentes de zero, pode-se aplicar o Método de Gauss.

In [16]:
```
linha_1 = np.array([2, 1, 3, 4])
linha_2 = np.array([1, 4, 2, 1])
linha_3 = np.array([3, 2, 1, 4])
linha_4 = np.array([2, 2, 3, 1])
```

In [17]:
```
def arredondar_matrix(matrix, casas_decimais):
    return Matrix(np.array(list(map(lambda x: round(x,casas_decimais), matrix))).reshape(matrix.shape))
```

In [18]:
```
matrix = [linha_1, linha_2, linha_3, linha_4]

def escalonamento(matrix):
    for index_pivo, linha_atual_pivo in enumerate(matrix):
        if index_pivo > len(matrix) - 2:
            break # Para se chegar a última linha
        for index, linha in enumerate(matrix):
            if index <= index_pivo:
                continue # Continua se for anterior ao pivo
            multiplicador = linha[index_pivo] / linha_atual_pivo[index_pivo]
            matrix[index] = linha - multiplicador * linha_atual_pivo
    return Matrix(matrix)
```

```
matriz = arredondar_matrix(escalonamento(matrix), 2)
matriz
```

Out[18]:

$$\begin{bmatrix} 2 & 1 & 3 & 4 \\ 0.0 & 3.5 & 0.5 & -1.0 \\ 0.0 & 0.0 & -3.57 & -1.86 \\ 0.0 & 0.0 & 0.0 & -2.64 \end{bmatrix}$$

Portanto, pelo Método de Gauss transformou a matriz em uma triangular superior: $\begin{pmatrix} 2 & 1 & 3 & 4 \\ 0 & 3,5 & 0,5 & -1 \\ 0 & 0 & -3,57 & -1,86 \\ 0 & 0 & 0 & -2,64 \end{pmatrix}$

In [19]:

```
solve(Eq(matriz * linha_x , linha_r), [x1, x2, x3, x4])
```

Out[19]:

```
{x1: 18.0440893542274,
 x2: 0.231800809603606,
 x3: -0.713038170501781,
 x4: -4.54521625163827}
```

Resolvendo o sistema linear, tem-se que $x_1 = 18,04; x_2 = 0,23; x_3 = -0,71; x_4 = -4,55$

Determinante da Matriz A = 2 x $3,5$ x $-3,57$ x $-2,64$ = $65,97$