1 Métodos Diretos para Solução de Sistemas de Equações Lineares

Nesse capítulo, estudaremos basicamente dois métodos diretos: o método da eliminação gaussiana e o método da decomposição LU. São métodos diretos e, por definição, podemos estimar qual o custo computacional uma vez que conseguimos calcular a quantidade de operações envolvidas. Tipicamente, apresentam desempenho bastante satisfatório.

1.1 Método da Eliminação Gaussiana

O *Método da Eliminação Gaussiana*, também referido como *método de Gauss* ou como *escalonamento* consiste em manipular o sistema de equações através de determinadas operações elementares, transformando a matriz estendida do sistema em uma matriz trapezoidal (chamada de *matriz escalonada do sistema*).

A **matriz estendida** (também chamada de *matriz completa*) de um sistema como $Ax = b$ é obtida acrescentando-se o vetor de termos independentes como última coluna, $[A|b]$, isto é,

$$[A|b] = \left[\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} & b_m \end{array}\right]. \tag{1}$$

A matriz escalonada, do tipo trapezoidal, será uma matriz cuja parte referente à matriz de coeficientes A se transforme em triangular superior, ou seja, o método da eliminação gaussiana promove uma triangularização (ou escalonamento) do sistema, como ilustrado abaixo:

$$\underbrace{\left[\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} & b_m \end{array}\right]}_{\text{matriz estendida}} \Rightarrow \underbrace{\left[\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ 0 & a'_{22} & \cdots & a'_{2n} & b'_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & a'_{mn} & b'_m \end{array}\right]}_{\text{matriz escalonada}} \tag{2}$$

Uma vez escalonado o sistema, a solução pode ser obtida via substituição retroativa, pois o sistema agora é triangular.

Naturalmente estas operações elementares devem preservar a solução do sistema e consistem em:

1. multiplicação de um linha por uma constante não nula.
2. substituição de uma linha por ela mesma somada a um múltiplo de outra linha.
3. permutação de duas linhas.

De uma maneira geral, o método segue as seguintes etapas:

1. **Construção da matriz estendida:**

$$[A|b] = \left[\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} & b_m \end{array}\right]. \tag{3}$$

2. **Eliminação dos termos da primeira coluna, a partir da segunda linha:** todos os termos referentes à incógnita x_1 são anulados, exceto o termo a_{11}, que é o *pivô* da coluna, ou seja, o elemento a partir do qual as operações elementares são conduzidas. A escolha do pivô, nesse formato, é basicamente pela posição (o primeiro da coluna). Em resumo, o objetivo é fazer $a_{i1} = 0$, para $i > 1$.

   Para isso, serão realizadas as seguintes substituições nas linhas da matriz (por notação, chamaremos aqui a i-ésima linha da matriz estendida de L_i, e a sua versão, após as alterações, de L_i'):

$$L_2' = L_2 - \frac{a_{21}}{a_{11}} \cdot L_1 \tag{4}$$

$$L_3' = L_3 - \frac{a_{31}}{a_{11}} \cdot L_1 \tag{5}$$

$$L_4' = L_4 - \frac{a_{41}}{a_{11}} \cdot L_1 \tag{6}$$

$$\vdots \tag{7}$$

$$L_m' = L_m - \frac{a_{m1}}{a_{11}} \cdot L_1 \tag{8}$$

   obtendo a seguinte matriz resultante

$$[A|b] = \left[\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ 0 & a_{22}' & \cdots & a_{2n}' & b_2' \\ 0 & a_{32}' & \cdots & a_{3n}' & b_3' \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & a_{m2}' & \cdots & a_{mn}' & b_m' \end{array}\right]. \tag{9}$$

3. **Eliminação dos termos da segunda coluna, a partir da terceira linha:** mesmo procedimento anterior, porém com o pivô sendo o elemento a_{22}, ou seja, o objetivo é fazer $a_{i2} = 0$, para $i > 2$.

   As substituições a serem feitas são:

$$L_3'' = L_3' - \frac{a_{32}}{a_{22}} \cdot L_2'$$

$$L_4'' = L_4' - \frac{a_{42}}{a_{22}} \cdot L_2'$$

$$\vdots$$

$$L_m'' = L_m' - \frac{a_{m2}}{a_{22}} \cdot L_2'$$

obtendo a seguinte matriz resultante

$$[A|b] = \left[\begin{array}{ccccc|c} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} & b_1 \\ 0 & a_{22}' & a_{23}' & \cdots & a_{2n}' & b_2' \\ 0 & 0 & a_{33}'' & \cdots & a_{3n}'' & b_3'' \\ 0 & 0 & a_{43}'' & \cdots & a_{4n}'' & b_4'' \\ \vdots & \vdots & \ddots & \vdots & \vdots & \\ 0 & 0 & a_{m3}'' & \cdots & a_{mn}'' & b_m'' \end{array}\right]. \tag{10}$$

---

4. **Obtenção da matriz escalonada:** os passos acima continuam até que a última linha tenha somente o elemento de maior índice, ou seja, que a última linha do sistema seja $a_{mn}x_n = b_n$, gerando a matriz escalonada na forma

$$\left[\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ 0 & a_{22}' & \cdots & a_{2n}' & b_2' \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & a_{mn}' & b_m' \end{array}\right] \tag{11}$$

Todos os coeficientes e termos independentes serão diferentes dos originais, exceto, talvez, na primeira linha.

5. **Cálculo da solução do sistema:** uma vez escalonada a matriz, basta, então, aplicar a técnica da substituição retroativa usada para sistemas triangulares superiores para encontrar os valores das incógnitas do sistema de equações lineares avaliado.

A função abaixo é uma implementação que mostra todas as saídas do processo de escalonamento e calcula o resultado do sistema no final.

In [1]:

```python
def elimin_gaussiana(a,b,imp = False, norma='inf'):
    import numpy as np #colocadas aqui para evitar problemas de namespace
    import scipy.linalg as sla #colocadas aqui para evitar problemas de namespace
    #matriz estendida Me
    b = np.mat(b).T
    Me = np.concatenate((a,b), axis=1)
    if imp: print('Matriz estendida: \n',Me,'\n\n')
    if imp: print('='*80)
    for j in range(Me.shape[-1] - 2): #coluna
        if imp: print('Operação na coluna', j)
        if imp: print('='*80, '\n\n')
        for i in range(j+1, Me.shape[0]): #linha
            Me[i] = Me[i] - (Me[i,j]/Me[j,j])*Me[j] #operações com cada linha
            if imp: print('Operação na linha', i, ':',Me[i], '\n')
            if imp: print('Matriz escalonando: \n',Me,'\n')
        if imp: print('='*80)
    if imp: print('\nMatriz escalonada: \n',Me,'\n')
    Matriz_A = np.delete(Me, -1, axis=1) #matriz de coeficientes escalonada
    Vetor_b = Me[:,-1] #matriz de termos independentes escalonada
    Vetor_x = sla.solve_triangular(Matriz_A, Vetor_b) # solução do sistema escalona
    if imp: print('='*80,'\n\nSolução do sistema: \n', Vetor_x)
    #verificação do erro do sistema
    if norma is 'inf': norma = np.inf
    if norma is '-inf': norma = -np.inf
    Erro = sla.norm(b - a.dot(Vetor_x),ord = norma)
    return (Matriz_A, Vetor_b, Vetor_x, Erro)
```

A função possui os seguintes parâmetros de entrada:

- **a:** matriz de coeficientes do sistema
- **b:** vetor de termos independentes (formato de vetor linha)
- **imp:** booleana, se `True` , cada passo do escalonamento é impresso na tela, se `False` , valor *default*, a impressão não aparece.
- **norma:** valores das normas usadas para calcular o erro da solução encontrada pela função. O valor `'inf'` é o *default*, equivalendo à norma infinita. Para outros valores, ver a documentação da função [norm (https://docs.scipy.org/doc/scipy/reference/generated/scipy.linalg.norm.html)](https://docs.scipy.org/doc/scipy/reference/generated/scipy.linalg.norm.html) do `scipy` .

retornando a tupla `(Matriz_A, Vetor_b, Vetor_x, Erro)` , em que:

- **Matriz_A:** matriz de coeficientes do sistema escalonado
- **Vetor_b:** vetor de termos independentes do sistema escalonado
- **Vetor_x:** vetor solução do sistema
- **Erro:** erro obtido pelo cálculo de norma do vetor resíduo $\mathbf{R} = \mathbf{b} - \mathbf{Ax}$.

No `scipy` , a função `solve` do módulo `linalg` implementa a solução de sistemas usando eliminação gaussiana. A documentação da função é mostrada [aqui (https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.linalg.solve.html)](https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.linalg.solve.html).

Os exemplos a seguir ilustram o uso do método.

Example 1 Resolva o sistema

$$
\begin{aligned}
2x_1 + 3x_2 - x_3 &= 5 \\
4x_1 + 4x_2 - 3x_3 &= 3 \\
2x_1 - 3x_2 + x_3 &= -1
\end{aligned} \tag{12}
$$

pelo método de eliminação gaussiana.

Resolução:

A matriz estendida do sistema é escrita como

$$
\begin{bmatrix} 2 & 3 & -1 & 5 \\ 4 & 4 & -3 & 3 \\ 2 & -3 & 1 & -1 \end{bmatrix}. \tag{13}
$$

Para anular os elementos da primeira coluna, abaixo do pivô a_{11}, realizaremos as seguintes alterações

$$
L_2' = L_2 - \frac{a_{21}}{a_{11}} \cdot L_1 = \begin{bmatrix} 4 & 4 & -3 & 3 \end{bmatrix} - \frac{4}{2}\begin{bmatrix} 2 & 3 & -1 & 5 \end{bmatrix} = \begin{bmatrix} 0 & -2 & -1 & -7 \end{bmatrix} \tag{14}
$$

$$
L_3' = L_3 - \frac{a_{31}}{a_{11}} \cdot L_1 = \begin{bmatrix} 2 & -3 & 1 & -1 \end{bmatrix} - \frac{2}{2}\begin{bmatrix} 2 & 3 & -1 & 5 \end{bmatrix} = \begin{bmatrix} 0 & -6 & 2 & -6 \end{bmatrix}, \tag{15}
$$

reescrevendo a matriz (13) como

$$
\begin{bmatrix} 2 & 3 & -1 & 5 \\ 0 & -2 & -1 & -7 \\ 0 & -6 & 2 & -6 \end{bmatrix}. \tag{16}
$$

Continuando o processo, agora tendo a_{22} como pivô da segunda coluna, realizaremos mais uma operação na linha 3 da matriz (16)

$$
L_3'' = L_3' - \frac{a_{32}}{a_{22}} \cdot L_2' = \begin{bmatrix} 0 & -6 & 2 & -6 \end{bmatrix} - \frac{-6}{-2}\begin{bmatrix} 0 & -2 & -1 & -7 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 5 & 15 \end{bmatrix}, \tag{17}
$$

obtendo, assim, a matriz escalonada

$$
\begin{bmatrix} 2 & 3 & -1 & 5 \\ 0 & -2 & -1 & -7 \\ 0 & 0 & 5 & 15 \end{bmatrix}, \tag{18}
$$

que corresponde ao sistema escalonado, que é triangular superior

$$
\begin{aligned}
2x_1 + 3x_2 - x_3 &= 5 \\
-2x_2 - x_3 &= -7 \\
5x_3 &= 15
\end{aligned} \tag{19}
$$

Aplicando, então, substituição reatroativa ao sistema escalonado, obtemos:

$$x_1 = 1 \qquad x_2 = 2 \qquad x_3 = 3. \tag{20}$$

Usando a função implementada, temos:

In [4]:

```
import numpy as np
import scipy.linalg as sla
```

In [12]:

```
a = np.array([[2,3,-1],[4,4,-3],[2,-3,1]])
b = np.array([5,3,-1])
M = elimin_gaussiana(a,b)
M
```

Out[12]:

```
(matrix([[ 2,  3, -1],
         [ 0, -2, -1],
         [ 0,  0,  5]]), matrix([[ 5],
         [-7],
         [15]]), array([[1.],
        [2.],
        [3.]]), 0.0)
```

Usando a função `solve`, obtemos:

In [4]:

```
sla.solve(a,b)
```

Out[4]:

```
array([1., 2., 3.])
```

Example 2 Resolva o sistema

$$\begin{aligned} x + y + z &= 1 \\ 4x + 4y + 2z &= 2 \\ 2x + y - z &= 0 \end{aligned} \tag{21}$$

pelo método de eliminação gaussiana.

Resolução:

A matriz estendida do sistema é

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 4 & 4 & 2 & 2 \\ 2 & 1 & -1 & 0 \end{bmatrix}. \tag{22}$$

Para anular os elementos da primeira coluna, abaixo do pivô a_{11}, realizaremos as seguintes alterações

$$L_2' = L_2 - \frac{a_{21}}{a_{11}} \cdot L_1 = \begin{bmatrix} 4 & 4 & 2 & 2 \end{bmatrix} - \frac{4}{1}\begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & -2 & -2 \end{bmatrix} \tag{23}$$

$$L_3' = L_3 - \frac{a_{31}}{a_{11}} \cdot L_1 = \begin{bmatrix} 2 & 1 & -1 & 0 \end{bmatrix} - \frac{2}{1}\begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & -3 & -2 \end{bmatrix}, \tag{24}$$

reescrevendo a matriz (22) como

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & -2 & -2 \\ 0 & -1 & -3 & -2. \end{bmatrix} \tag{25}$$

A seguir, ao invés de usar as operações acima, basta fazermos a permuta de posição entre a segunda e a terceira linhas de (25), obtendo

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & -1 & -3 & -2 \\ 0 & 0 & -2 & -2 \end{bmatrix} \tag{26}$$

que já é a matriz escalonada, correspondente ao sistema escalonado

$$\begin{aligned} x + y + z &= 1 \\ -y - 3z &= -2 \\ -2z &= -2. \end{aligned} \tag{27}$$

Aplicando, então, substituição reatroativa ao sistema escalonado, obtemos

$$x = 1 \qquad y = -1 \qquad z = 1. \tag{28}$$

Ao usarmos a função `solve` do `scipy` , obtemos:

In [5]:

```
a = np.array([[1,1,1],[4,4,2],[2,1,-1]])
b = np.array([1,2,0])
```

In [6]:

```
sla.solve(a,b)
```

Out[6]:

```
array([ 1., -1.,  1.])
```

Porém, ao usarmos a função `elimin_gaussiana` , obtemos:

In [8]:

```
elimin_gaussiana(a,b)
```

```
/opt/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:13:
RuntimeWarning: divide by zero encountered in long_scalars
  del sys.path[0]
/opt/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:13:
RuntimeWarning: invalid value encountered in multiply
  del sys.path[0]

---------------------------------------------------------------------
------
LinAlgError                               Traceback (most recent call
 last)
<ipython-input-8-bd3f1f9c6e1a> in <module>()
----> 1 elimin_gaussiana(a,b)

<ipython-input-1-0ebeb53ca982> in elimin_gaussiana(a, b, imp, norma)
     18     Matriz_A = np.delete(Me, -1, axis=1) #matriz de coeficien
tes escalonada
     19     Vetor_b = Me[:,-1] #matriz de termos independentes escalo
nada
---> 20     Vetor_x = sla.solve_triangular(Matriz_A, Vetor_b) # soluç
ão do sistema escalonado
     21     if imp: print('='*80,'\n\nSolução do sistema: \n', Vetor_
x)
     22     #verificação do erro do sistema

/opt/anaconda3/lib/python3.6/site-packages/scipy/linalg/basic.py in s
olve_triangular(a, b, trans, lower, unit_diagonal, overwrite_b, debu
g, check_finite)
    351     if info > 0:
    352         raise LinAlgError("singular matrix: resolution failed
 at diagonal %d" %
--> 353                           (info-1))
    354     raise ValueError('illegal value in %d-th argument of inte
rnal trtrs' %
    355                      (-info))

LinAlgError: singular matrix: resolution failed at diagonal 1
```

Exercise 1 Interprete o erro apresentado acima e altere a função **elimin_gaussiana**, para que esse erro não ocorra.

Essa abordagem de escolha do pivô como o elemento da coluna onde se realizará as operações que está na diagonal principal da matriz de coeficientes deixa o processo bastante sensível a propagação de erros durante as operações.

O exemplo a seguir ilustra bem isso.

Example 3 Resolva o sistema

$$\begin{aligned} x_1 + 4x_2 + 52x_3 &= 57 \\ 27x_1 + 110x_2 - 3x_3 &= 134 \\ 22x_1 + 2x_2 + 14x_3 &= 38 \end{aligned} \tag{29}$$

pelo método de eliminação gaussiana.

Resolução:

A matriz estendida do sistema é escrita como

$$\begin{bmatrix} 1 & 4 & 52 & 57 \\ 27 & 110 & -3 & 134 \\ 22 & 2 & 14 & 38 \end{bmatrix}. \tag{30}$$

Para anular os elementos da primeira coluna, abaixo do pivô a_{11}, realizaremos as seguintes alterações

$$L_2' = L_2 - \frac{a_{21}}{a_{11}} \cdot L_1 = \begin{bmatrix} 27 & 110 & -3 & 134 \end{bmatrix} - \frac{27}{1}\begin{bmatrix} 1 & 4 & 52 & 57 \end{bmatrix} = \begin{bmatrix} 0 & 2 & -1400 & -1 \end{bmatrix}$$

$$L_3' = L_3 - \frac{a_{31}}{a_{11}} \cdot L_1 = \begin{bmatrix} 22 & 2 & 14 & 38 \end{bmatrix} - \frac{22}{1}\begin{bmatrix} 1 & 4 & 52 & 57 \end{bmatrix} = \begin{bmatrix} 0 & -86 & -1130 & -12 \end{bmatrix}$$

reescrevendo a matriz (30) como

$$\begin{bmatrix} 2 & 3 & -1 & 5 \\ 0 & 2 & -1400 & -1410 \\ 0 & -86 & -1130 & -1210 \end{bmatrix}. \tag{33}$$

Continuando o processo,

$$\begin{aligned} L_3'' = L_3' - \frac{a_{32}}{a_{22}} \cdot L_2' &= \begin{bmatrix} 0 & -86 & -1130 & -1210 \end{bmatrix} - \frac{-86}{2}\begin{bmatrix} 0 & 2 & -1400 & -1410 \end{bmatrix} \\ &= \begin{bmatrix} 0 & 0 & -61300 & -61800 \end{bmatrix}, \end{aligned} \tag{34}$$

obtendo, assim, a matriz escalonada

$$\begin{bmatrix} 1 & 4 & 52 & 57 \\ 0 & 2 & -1400 & -1410 \\ 0 & 0 & -61300 & -61800 \end{bmatrix}, \tag{35}$$

que corresponde ao sistema escalonado, que é triangular superior

$$\begin{aligned} x_1 + 4x_2 + 52x_3 &= 57 \\ 2x_2 - 1400x_3 &= -1410 \\ -61300x_3 &= -61800 \end{aligned} \tag{36}$$

Aplicando, então, substituição reatroativa ao sistema escalonado, obtemos

$$x_1 = 4.5 \qquad x_2 = 0 \qquad x_3 = 1.01, \tag{37}$$

bastante diferente de

$$x_1 = 1 \qquad x_2 = 1 \qquad x_3 = 1, \tag{38}$$

a solução exata do sistema.

Exercise 2 Quantifique o erro da solução encontrada acima.

Esse exemplo ilustra bem os efeitos de propagação de erros quando escalonamos matrizes usando o método de eliminação gaussiana convencional.

Esse efeito pode ser mensurado melhor quando consideramos o problema com elementos com grande diferença de escala, como a seguir.

Existem duas alternativas à essa escolha de pivô por posição:

- **pivotamento parcial**: o pivô escolhido é o maior número na coluna a ser avaliada;
- **pivotamento total**: o pivô escolhido é o maior número na matriz de coeficientes (ver livro do Chapra).

1.1.1 Eliminação gaussiana com pivotamento parcial

A eliminação gaussiana com **pivotamento parcial** consiste em fazer uma permutação de linhas de forma a escolher o maior pivô (em módulo) a cada passo.

Os exemplos a seguir ilustram o uso da técnica.

Example 4 Resolva o sistema

$$\begin{aligned} 0.02x_1 + 0.01x_2 &= 0.02 \\ x_1 + 2x_2 + x_3 &= 1 \\ x_2 + 2x_3 + x_4 &= 4 \\ 100x_3 + 200x_4 &= 800 \end{aligned} \tag{39}$$

por eliminação gaussiana com pivotamento parcial.

Resolução:

A matriz estendida do sistema é dada por

$$\left[\begin{array}{cccc|c} 0.02 & 0.01 & 0 & 0 & 0.02 \\ 1 & 2 & 1 & 0 & 1 \\ 0 & 1 & 2 & 1 & 4 \\ 0 & 0 & 100 & 200 & 800 \end{array}\right]$$

Olhando para a primeira coluna, vê-se que o pivô é o elemento $a_{21} = 1$, maior elemento da coluna. Assim, precisamos trocar as linhas 1 e 2 de posição, para que o pivô fique sempre como primeiro elemento da coluna a ser escalonada. Assim,

$$\left[\begin{array}{cccc|c} 0.02 & 0.01 & 0 & 0 & 0.02 \\ 1 & 2 & 1 & 0 & 1 \\ 0 & 1 & 2 & 1 & 4 \\ 0 & 0 & 100 & 200 & 800 \end{array}\right]$$

$$\downarrow$$

$$\left[\begin{array}{cccc|c} 1 & 2 & 1 & 0 & 1 \\ 0.02 & 0.01 & 0 & 0 & 0.02 \\ 0 & 1 & 2 & 1 & 4 \\ 0 & 0 & 100 & 200 & 800 \end{array}\right]$$

Seguindo o procedimento de escalonamento do método de eliminação de Gauss, obtemos

$$\left[\begin{array}{cccc|c} 1 & 2 & 1 & 0 & 1 \\ 0.02 & 0.01 & 0 & 0 & 0.02 \\ 0 & 1 & 2 & 1 & 4 \\ 0 & 0 & 100 & 200 & 800 \end{array}\right]$$

$$\downarrow$$

$$\left[\begin{array}{cccc|c} 1 & 2 & 1 & 0 & 1 \\ 0 & -0.03 & -0.02 & 0 & 0 \\ 0 & 1 & 2 & 1 & 4 \\ 0 & 0 & 100 & 200 & 800 \end{array}\right]$$

Olhando para a segunda coluna, considerando as linhas 2 abaixo, escolhemos um novo pivô, nesse caso o elemento $a_{23} = 1$, maior elemento da segunda linha pra baixo da segunda coluna, obtendo

$$\left[\begin{array}{cccc|c} 1 & 2 & 1 & 0 & 1 \\ 0 & -0.03 & -0.02 & 0 & 0 \\ 0 & 1 & 2 & 1 & 4 \\ 0 & 0 & 100 & 200 & 800 \end{array}\right]$$

$$\downarrow$$

$$\left[\begin{array}{cccc|c} 1 & 2 & 1 & 0 & 1 \\ 0 & 1 & 2 & 1 & 4 \\ 0 & -0.03 & -0.02 & 0 & 0 \\ 0 & 0 & 100 & 200 & 800 \end{array}\right]$$

e, pela eliminação gaussiana,

$$\left[\begin{array}{cccc|c} 1 & 2 & 1 & 0 & 1 \\ 0 & 1 & 2 & 1 & 4 \\ 0 & -0.03 & -0.02 & 0 & 0 \\ 0 & 0 & 100 & 200 & 800 \end{array}\right]$$

$$\downarrow$$

$$\left[\begin{array}{cccc|c} 1 & 2 & 1 & 0 & 1 \\ 0 & 1 & 2 & 1 & 4 \\ 0 & 0 & 0.04 & 0.03 & 0.12 \\ 0 & 0 & 100 & 200 & 800 \end{array}\right]$$

Analisando agora a coluna 3, a partir da linha 3, temos que o novo pivô será $a_{34} = 100$, então

$$\left[\begin{array}{cccc|c} 1 & 2 & 1 & 0 & 1 \\ 0 & 1 & 2 & 1 & 4 \\ 0 & 0 & 0.04 & 0.03 & 0.12 \\ 0 & 0 & 100 & 200 & 800 \end{array}\right]$$

$$\downarrow$$

$$\left[\begin{array}{cccc|c} 1 & 2 & 1 & 0 & 1 \\ 0 & 1 & 2 & 1 & 4 \\ 0 & 0 & 100 & 200 & 800 \\ 0 & 0 & 0.04 & 0.03 & 0.12 \end{array}\right]$$

e aplicando a eliminação de gauss, obtém-se

$$\left[\begin{array}{cccc|c} 1 & 2 & 1 & 0 & 1 \\ 0 & 1 & 2 & 1 & 4 \\ 0 & 0 & 100 & 200 & 800 \\ 0 & 0 & 0.04 & 0.03 & 0.12 \end{array}\right]$$

$$\downarrow$$

$$\left[\begin{array}{cccc|c} 1 & 2 & 1 & 0 & 1 \\ 0 & 1 & 2 & 1 & 4 \\ 0 & 0 & 100 & 200 & 800 \\ 0 & 0 & 0 & -0.05 & -0.2 \end{array}\right]$$

que é a matriz escalonada.

Por fim, por substituição retroativa, obtemos:

$$\begin{aligned} -0.2x_4 &= -0.05 & \Rightarrow \quad x_4 &= 4 \\ 100x_3 + 200x_4 &= 800 & \Rightarrow \quad x_3 &= 0 \\ x_2 + 2x_3 + x_4 &= 4 & \Rightarrow \quad x_2 &= 0 \\ x_1 + 2x_2 + x_3 &= 1 & \Rightarrow \quad x_1 &= 1 \end{aligned} \tag{40}$$

Example 5 Resolva o sistema

$$\begin{aligned} x + y + z &= 1 \\ 2x + y - z &= 0 \\ 2x + 2y + z &= 1 \end{aligned} \tag{41}$$

por eliminação gaussiana com pivotamento parcial.

Resolução:

A matriz estendida do sistema é

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & 0 \\ 2 & 2 & 1 & 1 \end{bmatrix} \tag{42}$$

O pivô escolhido é o elemento a_{12}, então são trocadas de posição as linhas 1 e 2, e segue-se o procedimento da eliminação gaussiana para a primeira coluna.

$$\begin{bmatrix} 2 & 1 & -1 & 0 \\ 1 & 1 & 1 & 1 \\ 2 & 2 & 1 & 1 \end{bmatrix} \sim \begin{bmatrix} 2 & 1 & -1 & 0 \\ 0 & 1/2 & 3/2 & 1 \\ 0 & 1 & 2 & 1 \end{bmatrix} \tag{43}$$

Após o procedimento para a primeira coluna, novamente o pivô escolhido na segunda coluna segue o mesmo critério, o de maior valor. Dessa forma, observamos que o pivô escolhido é o elemento a_{32}, então são trocadas de posição as linhas 2 e 3, e segue-se o procedimento da eliminação gaussiana para a segunda coluna.

$$\begin{bmatrix} 2 & 1 & -1 & 0 \\ 0 & 1 & 2 & 1 \\ 0 & 1/2 & 3/2 & 1 \end{bmatrix} \sim \begin{bmatrix} 2 & 1 & -1 & 0 \\ 0 & 1 & 2 & 1 \\ 0 & 0 & 1/2 & 1/2 \end{bmatrix} \tag{44}$$

Encontramos $1/2z = 1/2$, ou seja, $z = 1$. Substituímos na segunda equação e temos $y + 2z = 1$, ou seja, $y = -1$ e, finalmente $2x + y - z = 0$, resultando em $x = 1$.

Exercise 3 Refaça o exemplo 3 usando eliminação gaussiana com pivotamento parcial e compare os erros do resultado obtido aqui com os do exemplo.

Example 6 Resolva o seguinte sistema usando eliminação gaussiana sem e com pivotamento parcial, e discuta o resultado frente à aritmética de ponto flutuante quando $0 < |\epsilon| \ll 1$.

$$\begin{bmatrix} \varepsilon & 2 \\ 1 & \varepsilon \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 4 \\ 3 \end{bmatrix} \tag{45}$$

Resolução:

Vamos, primeiramente, executar a eliminação gaussiana sem pivotamento parcial para $\varepsilon \neq 0$ e $|\varepsilon| \ll 1$:

$$\underbrace{\left[\begin{array}{cc|c} \varepsilon & 2 & 4 \\ 1 & \varepsilon & 3 \end{array}\right]}_{\text{Matriz estendida}} \Rightarrow \underbrace{\left[\begin{array}{cc|c} \varepsilon & 2 & 4 \\ 0 & \varepsilon - \dfrac{2}{\varepsilon} & 3 - \dfrac{4}{\varepsilon} \end{array}\right]}_{\text{Matriz escalonada}}, \tag{46}$$

de onde podemos obter

$$y = \frac{3 - 4/\varepsilon}{\varepsilon - 2/\varepsilon} \tag{47}$$

e

$$x = \frac{4 - 2y}{\varepsilon}. \tag{48}$$

Observe que a expressão obtida para y se aproximada de 2 quando ε é pequeno ($\varepsilon \neq 0$ e $|\varepsilon| \ll 1$):

$$y = \frac{3 - 4/\varepsilon}{\varepsilon - 2/\varepsilon} = \frac{3\varepsilon - 4}{\varepsilon^2 - 2} \longrightarrow \frac{-4}{-2} = 2, \quad \text{quando } \varepsilon \to 0. \tag{49}$$

Já expressão obtida para x depende justamente da diferença $2 - y$:

$$x = \frac{4 - 2y}{\varepsilon} = \frac{2}{\varepsilon}(2 - y) \tag{50}$$

Assim, quando ε é pequeno, a primeira expressão, implementada em um sistema de ponto flutuante de acurácia finita, produz $y = 2$ e, consequentemente, a expressão para x produz $x = 0$. Isto é, estamos diante um problema de cancelamento catastrófico (https://pt.wikipedia.org/wiki/Cancelamento_catastr%C3%B3fico).

Agora, quando usamos a eliminação gaussiana com pivotamento parcial, fazemos uma permutação de linhas de forma a escolher o maior pivô a cada passo:

$$\left[\begin{array}{cc|c} \varepsilon & 2 & 4 \\ 1 & \varepsilon & 3 \end{array}\right] \sim \left[\begin{array}{cc|c} 1 & \varepsilon & 3 \\ \varepsilon & 2 & 4 \end{array}\right] \sim \left[\begin{array}{cc|c} 1 & \varepsilon & 3 \\ 0 & 2-\varepsilon^2 & 4-3\varepsilon \end{array}\right] \tag{51}$$

Continuando o procedimento, temos:

$$y = \frac{4-4\varepsilon}{2-\varepsilon^2} \tag{52}$$

e

$$x = 3 - \varepsilon y \tag{53}$$

Observe que tais expressões são analiticamente idênticas às anteriores, no entanto, são mais estáveis numericamente. Quando ε converge a zero, y converge a 2, como no caso anterior. No entanto, mesmo que $y = 2$, a segunda expressão produz $x = 3 - \varepsilon y$, isto é, a aproximação $x \approx 3$ não depende mais de obter $2 - y$ com precisão.

Exercise 4 Implemente uma versão da função **elimin_gaussiana** que faça o pivotamento parcial. Compare os resultados e erros com os exemplos anteriores.

Exercise 5 Verifique o exemplo 5 para valores pequenos de ε, usando as funções com e sem pivotamento parcial.

1.2 Fatoração LU

Considere um sistema linear $Ax = b$, onde a matriz A é densa (diferentemente de uma matriz esparsa, uma matriz densa possui a maioria dos elementos diferentes de zero). A fim de resolver o sistema, podemos fatorar a matriz A como o produto de uma matriz L triangular inferior, com diagonal unitária e uma matriz U triangular superior, ou seja, fazer $A = LU$, num procedimento conhecido, obviamente, como fatoração LU.

A fatoração LU é, geralmente, utilizada quando precisamos resolver vários sistemas de equações lineares, sendo todos compostos pela mesma matriz dos coeficientes A.

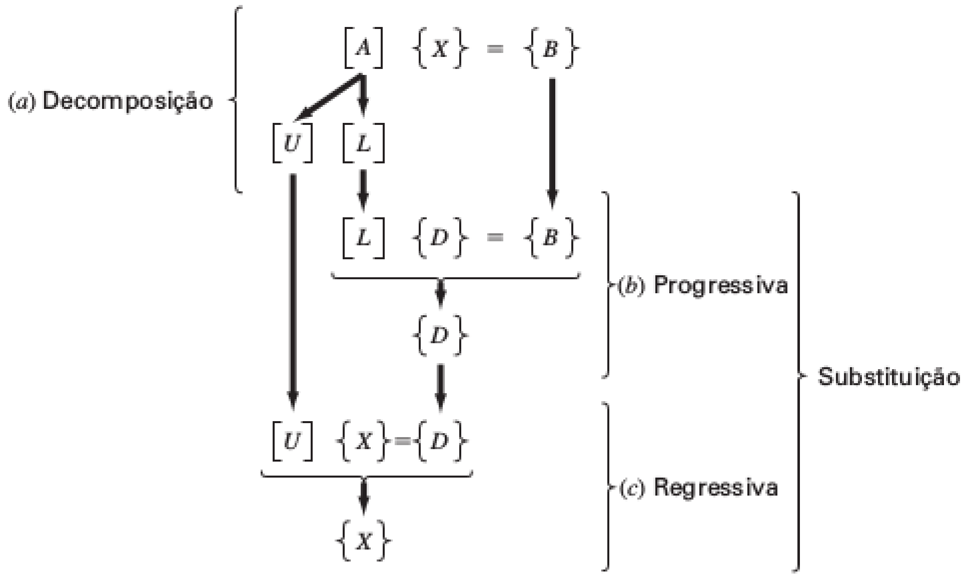
Assim, para uma matriz de coeficientes 3×3, por exemplo, temos

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix} \tag{54}$$

Sendo assim, o sistema a ser resolvido pode ser reescrito da seguinte forma:

$$Ax = b \tag{55}$$
$$(LU)x = b \tag{56}$$
$$L(Ux) = b \tag{57}$$
$$Ly = b \text{ e } \ Ux = y \tag{58}$$

Isto significa que, ao invés de resolvermos o sistema original, podemos resolver o sistema triangular inferior $Ly = b$ e, então, o sistema triangular superior $Ux = y$, o qual nos fornece a solução de $Ax = b$. Os passos da decomposição são ilustrados na figura abaixo.



A matriz U da fatoração (não vamos usar pivotamento nesse primeiro exemplo) LU é a matriz obtida ao final do escalonamento da matriz A, triangular superior.

A matriz L é construída a partir da matriz identidade I (diagonal unitária, demais elementos nulos), ao longo do escalonamento de A. Os elementos da matriz L são os multiplicadores do primeiro elemento da linha de A a ser zerado dividido pelo pivô acima na mesma coluna,

$$l_{ij} = \frac{a_{ij}}{a_{jj}}. \tag{59}$$

Example 7 Use a fatoração LU para resolver o seguinte sistema linear:

$$\begin{aligned} x_1 + x_2 + x_3 &= -2 \\ 2x_1 + x_2 - x_3 &= 1 \\ 2x_1 - x_2 + x_3 &= 3 \end{aligned} \tag{60}$$

Resolução:

Começamos fatorando a matriz A dos coeficientes deste sistema:

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 1 & -1 \\ 2 & -1 & 1 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{I_{3,3}} \underbrace{\begin{bmatrix} 1 & 1 & 1 \\ 2 & 1 & -1 \\ 2 & -1 & 1 \end{bmatrix}}_{A} \tag{61}$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ \frac{a_{21}}{a_{11}} = 2 & 1 & 0 \\ \frac{a_{31}}{a_{11}} = 2 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 0 & -1 & -3 \\ 0 & -3 & -1 \end{bmatrix} \tag{62}$$

$$= \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 2 & \frac{a_{32}}{a_{22}} = 3 & 1 \end{bmatrix}}_{L} \underbrace{\begin{bmatrix} 1 & 1 & 1 \\ 0 & -1 & -3 \\ 0 & 0 & 8 \end{bmatrix}}_{U} \tag{63}$$

Completada a fatoração LU, resolvemos, primeiramente, o sistema $Ly = b$:

$$
\begin{aligned}
y_1 &= -2 \\
2y_1 + y_2 &= 1 \\
2y_1 + 3y_2 + y_3 &= 3
\end{aligned} \tag{64}
$$

o qual nos fornece $y_1 = -2$, $y_2 = 5$ e $y_3 = -8$. Por fim, obtemos a solução resolvendo o sistema $Ux = y$:

$$
\begin{aligned}
x_1 + x_2 + x_3 &= -2 \\
-x_2 - 3x_3 &= 5 \\
8x_3 &= -8
\end{aligned} \tag{65}
$$

o qual fornece $x_3 = -1$, $x_2 = -2$ e $x_1 = 1$.

1.2.1 Decomposição LU usando Python

A biblioteca `scipy` possui três funções, no submódulo `linalg`, para decomposição LU e resolução de sistemas lineares:

- **lu:** faz a decomposição LU de uma matriz A, decompondo no formato $A = PLU$. Nessa função, a implementação é feita usando somente o scipy. Esse método retorna, se `permute_l == False`, a matriz de permutação P, a matriz L e a matriz U, e se `permute_l == True`, retorna a matriz PL e a matriz U. Mais informações na [documentação (https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.linalg.lu.html)](https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.linalg.lu.html);

- **lu_factor:** faz a decomposição LU de uma matriz A, decompondo no formato $A = PLU$. Porém, essa função, implementa o método `*GETRF` da `LAPACK`. Esse método retorna uma matriz LU, contendo a matriz U no triângulo superior e a matriz L no inferior e um *array* `piv` contendo os índices dos pivôs, representando a matriz de permutação. Mais informações na [documentação (https://docs.scipy.org/doc/scipy/reference/generated/scipy.linalg.lu_factor.html)](https://docs.scipy.org/doc/scipy/reference/generated/scipy.linalg.lu_factor.html);

- **lu_solve:** resolve um sistema do tipo $Ax = B$, a partir da fatorização LU. Como entrada, recebe as matrizes LU e piv obtidas da função `lu_factor` e o vetor B, retornando um *array* x com a solução do sistema. Mais informações na [documentação (https://docs.scipy.org/doc/scipy/reference/generated/scipy.linalg.lu_solve.html)](https://docs.scipy.org/doc/scipy/reference/generated/scipy.linalg.lu_solve.html).

Vamos resolver o exemplo anterior usando python.

In [15]:

```
A = np.array([[1,1,1],[2,1,-1],[2,-1,1]])
B = np.array([-2,1,3])
A
```

Out[15]:

```
array([[ 1,  1,  1],
       [ 2,  1, -1],
       [ 2, -1,  1]])
```

Se quisermos saber quem são as matrizes L e U, usamos:

In [21]:

```
P,L,U = sla.lu(A)
print(P)
print(L)
print(U)
```

```
[[0. 0. 1.]
 [1. 0. 0.]
 [0. 1. 0.]]
[[ 1.    0.    0.  ]
 [ 1.    1.    0.  ]
 [ 0.5  -0.25  1.  ]]
[[ 2.  1. -1.]
 [ 0. -2.  2.]
 [ 0.  0.  2.]]
```

Calculando usando `lu_factor` , obtemos:

In [22]:

```
lu, piv = sla.lu_factor(A)
print(lu)

print(piv)
```

```
[[ 2.    1.   -1.  ]
 [ 1.   -2.    2.  ]
 [ 0.5  -0.25  2.  ]]
[1 2 2]
```

Resolvendo o sistema, obtemos:

In [23]:

```
sla.lu_solve((lu, piv), B)
```

Out[23]:

```
array([ 1., -2., -1.])
```

Exercise 6 Note que os resultados obtidos para as matrizes L e U são diferentes do calculado pelo python e no método, porém, o resultado do sistema de equações confere entre os dois casos. Explique a diferença desses resultados (veja a diferença entre $A = LU$ e $A = PLU$).