P2 - Jacobi - Seidel

Data da entrega : 02/07 até as 15:00hs

- Complete o código (marcado com `None`) e escreve textos diretamente nos notebooks.
- Execute todo notebook e salve tudo em um PDF nomeado como "NomeSobrenome-P2.pdf"
- Envie o PDF via formulário : https://forms.gle/yRhiTMsjV281KkTw9

1 - Pacotes

Primeiro, vamos executar a célula abaixo para importar todos os pacotes que precisaremos.

- numpy é o pacote fundamental para a computação científica com Python.

In [4]:
```
import numpy as np
```

In [5]:
```
# Método de Jacobi

def jacobi(A, B):

    # Método de Jacobi (Gladston Moreira)
    """
    input A = nxn array
          B = nx1 array
    """
    # Verifica se o sistema é compatível
    d = np.linalg.det(A)
    if d == 0:
        print('Sistema não é compatível')

    # Estimativa Inicial
    n = len(A)
    X = np.zeros((n, 1)) # Criando uma matriz de zeros de tamanho Nx1
    Xk = np.copy(X)

    # Verifica condições de convergência
    for i in range(n):

        L = 0 # Armazena a soma de todos os não-pivôs de uma determinada linha i
        C = 0 # Armazena a soma de todos os não-pivôs de uma determinada coluna j

        # Verificando as linhas devido a condição estabelecida para os pivôs na vari
        for j in range(n):
            if j != i: # Comparação para verificar se não é um pivô
                L += abs(A[i, j]) # Armazena informação

        # Verificando as colunas devido a condição estabelecida para os pivôs na var
        for k in range(n):
            if k != i: # Comparação para verificar se não é um pivô
                C += abs(A[k, i]) # Armazena informação

        if L > abs(A[i, i]):
            print('Linha', i+1, ' não satisfaz critério das linhas')
            break
        elif C > abs(A[i, i]):
```

```
            print('Coluna', i+1, ' não satisfaz critério das colunas')
            break

    # Critérios de parada
    maxiter = 10
    minDelta = 1e-09
    delta = 1

    # Contador
    k = 0
    print('\nk -\t X \t\t- \t\t max |X(k) - X(k-1)| ')
    print(k,'\t',np.transpose(X),'\t\t\t---')
    while k < maxiter and delta > minDelta:
        # Para cada linha
        for i in range(n):
            # Multiplica e soma elementos conhecidos e joga para o outro lado
            soma = B[i, 0]

            # Percorrendo cada um dos itens da linha
            for j in range(n):
                if i != j:
                    # Armazenando o valor das variáveis no momento k-1 em Xk.
                    soma += (A[i, j] * (-1)) * Xk[j, 0]

            soma /= A[i, i]
            # Recebendo valor da soma
            X[i, 0] = soma

        # Calcula max|x(i)k - x(i)k-1|
        aux = []
        for i in range(n):
            aux.append(abs(X[i, 0] - Xk[i, 0]))
        delta = max(aux)

        # Iteração
        k = k + 1

        # print k -- xk   --  max|x(i)k-x(i)k-1|
        print(k,'\t', np.transpose(X), '\t\t\t', delta)
        # Armazenando o valor das variáveis x no momento k-1 em Xk.
        Xk = np.copy(X)

    print('X:')
    print(X)

    return X
```

In [6]:

```
# Teste
A = np.array([[1,-1,-1],[1 ,-8, -1],[1, -1 ,-8]])
B=np.array([[2],[3],[0]])
X = jacobi(A,B)
```

```
Linha 1  não satisfaz critério das linhas

k -      X              -               max |X(k) - X(k-1)|
0        [[0. 0. 0.]]                  ---
1        [[ 2.    -0.375 -0.   ]]                    2.0
2        [[ 1.625    -0.125     0.296875]]                    0.375
3        [[ 2.171875   -0.20898438  0.21875   ]]                    0.546875
4        [[ 2.00976562 -0.13085938  0.29760742]]                    0.162109375
5        [[ 2.16674805 -0.16098022  0.26757812]]                    0.156982421
875
6        [[ 2.1065979  -0.13760376  0.29096603]]                    0.060150146
```

```
484375
7       [[ 2.15336227 -0.14804602  0.28052521]]                    0.046764373
779296875
8       [[ 2.13247919 -0.14089537  0.28767604]]                    0.020883083
34350586
9       [[ 2.14678067 -0.14439961  0.28417182]]                    0.014301478
862762451
10      [[ 2.13977221 -0.14217389  0.28639753]]                    0.007008455
693721771
X:
[[ 2.13977221]
 [-0.14217389]
 [ 0.28639753]]
```

In [7]:

```
def gaussseidel(A,B):

    # Método de GaussSeidel (Gladston Moreira)
    """
    input A = nxn array
          B = nx1 array
    """
    # Verifica se o sistema é compatível
    d = np.linalg.det(A)
    if d==0:
        print('Sistema não é compatível')

    # Estimativa Inicial
    n = len(A)
    X = np.zeros((n, 1)) # Criando uma matriz de zeros de tamanho Nx1
    Xk = np.copy(X)

    # Verifica condições de convergência
    for i in range(n):

        L = 0 # Armazena a soma de todos os não-pivôs de uma determinada linha i
        C = 0 # Armazena a soma de todos os não-pivôs de uma determinada coluna j

        # Verificando as linhas devido a condição estabelecida para os pivôs na vari
        for j in range(n):
            if j != i: # Comparação para verificar se não é um pivô
                L += abs(A[i, j]) # Armazena informação

        # Verificando as colunas devido a condição estabelecida para os pivôs na var
        for k in range(n):
            if k != i: # Comparação para verificar se não é um pivô
                C += abs(A[k, i]) # Armazena informação

        if L > abs(A[i, i]):
            print('Linha', i+1, ' não satisfaz critério das linhas')
            break
        elif C > abs(A[i, i]):
            print('Coluna', i+1, ' não satisfaz critério das colunas')
            break

    # Critérios de parada
    maxiter = 10
    minDelta = 1e-09
    delta = 1

    # Contador
    k = 0
    print('\nk -\t X \t\t- \t\t max |X(k) - X(k-1)| ')
    print(k,'\t', np.transpose(X), '\t\t\t---')
    while k < maxiter and delta > minDelta:
```

```
        # Para cada linha
        for i in range(n):
            soma = B[i, 0]

            # Numero de componentes atualizados
            num_atualizados = i

            # Multiplica e soma elementos conhecidos e joga para o outro lado
            # Percorrendo cada um dos itens da linha
            for j in range(n):
                # Armazenando o valor das variáveis no momento k-1 em Xk.
                if i != j:
                    if num_atualizados > 0:
                        soma += (A[i, j] * (-1)) * X[j, 0]
                        num_atualizados -= 1
                    else:
                        soma += (A[i, j] * (-1)) * Xk[j, 0]

            soma /= A[i, i]
            # Recebendo valor da soma
            X[i, 0] = soma

        # Calcula max|x(i)k-x(i)k-1|
        aux = [] # Armazena os valores de |x(i)k - x(i)k-1| para cada linha da matri
        for i in range(n):
            aux.append(abs(X[i, 0] - Xk[i, 0]))

        delta = max(aux)

        # interação
        k = k + 1

        # print k -- xk    --  max|x(i)k-x(i)k-1|
        print(k,'\t',np.transpose(X),'\t\t\t',delta)
        # Armazenando o valor das variáveis no momento k-1 em Xk.
        Xk = np.copy(X)

    print('X:')
    print(X)

    return X
```

In [8]:

```
# Teste
A = np.array([[1,-1,-1],[1 ,-8, -1],[1, -1 ,-8]])
B=np.array([[2],[3],[0]])
X = gaussseidel(A,B)
```

```
Linha 1  não satisfaz critério das linhas

k -      X              -               max |X(k) - X(k-1)|
0        [[0. 0. 0.]]                  ---
1        [[ 2.       -0.125     0.265625]]                    2.0
2        [[ 2.140625   -0.140625    0.28515625]]                    0.140625
3        [[ 2.14453125 -0.14257812  0.28588867]]                    0.00390625
4        [[ 2.14331055 -0.14282227  0.2857666 ]]                    0.001220703
125
5        [[ 2.14294434 -0.14285278  0.28572464]]                    0.000366210
9375
6        [[ 2.14287186 -0.1428566   0.28571606]]                    7.247924804
6875e-05
7        [[ 2.14285946 -0.14285707  0.28571457]]                    1.239776611
328125e-05
8        [[ 2.14285749 -0.14285713  0.28571433]]                    1.966953277
```

```
5878906e-06
9       [[ 2.14285719 -0.14285714  0.28571429]]                    2.980232238
769531e-07
10      [[ 2.14285715 -0.14285714  0.28571429]]                    4.377216100
692749e-08
X:
[[ 2.14285715]
 [-0.14285714]
 [ 0.28571429]]
```

In [ ]: