Escola Politécnica Da Universidade de São Paulo

MAP3121 - Métodos Numéricos e Aplicações

EP1

Aluno:
Felipe Cardenas Lima Namour
NUSP: 11807111

Aluno:
Brenda Moreira Santos
NUSP: 11384818

1 Introdução

O código escrtio durante esse trabalho tem como objetivo fazer a solução de sistemas lineares utilizando decomposição LU, além de implementar uma solução mais eficiente para matrizes cíclicas e tridiagonais. Ambas as implementações são escolhidas ao inciar a execução do programa, junto com a matriz a ser resolvida.

2 Testes e resultados

Para teste do código foi gerado uma matriz cíclica 20x20 utilizando as relações de construção para os vetores a,b e c descritas no enunciado do problema.

O resultado obtido para o vetor solução X:

[0.33031512
0.33369784
0.33082061
0.32458573
0.3105381
0.28498139
0.24375728
0.18349137
0.10274415
0.00360629
-0.10669724
-0.2147279
-0.30113746
-0.34330813
-0.32097501
-0.22451082
-0.0638644
0.12580676
0.28713644
0.35589205]

Alguns resultados intermediários do código são:

Vetor aT:

[0
0.375
0.41666667
0.4375
0.45
0.45833333
0.46428571
0.46875
0.47222222
0.475
0.47727273
0.47916667
0.48076923
0.48214286
0.48333333
0.484375
0.48529412
0.48611111
0.48684211]

Vetor bT:

[2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2]

Vetor cT:

[0.75
0.625
0.58333333
0.5625
0.55
0.54166667
0.53571429
0.53125
0.52777778
0.525
0.52272727
0.52083333
0.51923077
0.51785714
0.51666667
0.515625
0.51470588
0.51388889
0]

Vetor dT:

[9.99876632e-01
9.98026728e-01
9.90023658e-01
9.68583161e-01
9.23879533e-01
8.44327926e-01
7.18126298e-01
5.35826795e-01
2.94040325e-01
6.12323400e-17
-3.23917418e-01
-6.37423990e-01
-8.83765630e-01
-9.98026728e-01
-9.23879533e-01
-6.37423990e-01
-1.71929100e-01
3.68124553e-01
8.18149717e-01]

3 Código

As primeiras funções são destinadas a organização dos dados dentro do código. Sendo elas responsáveis por montarem as matrizes e organizarem elas conforme a entrada de dados do usuário, seja por matrizes, vetores ou pela descrição de matriz cíclia dada na tarefa do enunciado.

```
import numpy as np

def inputMatriz(n):
    matriz = np.zeros((n,n))
    for i in range(n):
        for j in range(n):
            print("De o valor " + str(i+1) + "," + str(j+1))
            matriz[i][j] = input()

    print("Matriz A:")
    print(str(matriz))

    #INPUT DO VETOR RESPOSTA
    print("Agora de o input do vetor resposta")
    d = np.zeros(n)
    for i in range(n):
        print("De o valor " + str(i + 1))
```

```
        d[i] = input()

    print("Vetor resposta;")
    print(str(d))

    #Separar vetores a, b e c
    a = np.zeros(n)
    b = np.zeros(n)
    c = np.zeros(n)

    a[0] = matriz[0][n-1]
    c[n-1] = matriz[n-1][0]
    for i in range(n):
        b[i] = matriz[i][i]
        if (i != (n-1)):
            a[i+1] = matriz[i+1][i]
        if (i != (n-1)):
            c[i] = matriz[i][i+1]

    print("Vetor A:")
    print(str(a))
    print("Vetor B:")
    print(str(b))
    print("Vetor C:")
    print(str(c))

    return a,b,c,d

def montaMatrizN(n): #[U+FFFD][U+FFFD]respons[U+FFFD]vel por gerar os vetores
    #a,b,c,d,bT,cT,v,w
    d  = np.zeros(n)
    a  = np.zeros(n)
    b  = np.zeros(n)
    c  = np.zeros(n)
    bT = np.zeros(n)
    cT = np.zeros(n)
    aT  = np.zeros(n)
    dT = np.zeros(n)
    v  = np.zeros(n-1)

    for i in range(n):
        if i == (n-1): #Gerar vetor a
            a[i] = ((2*n)-1)/(2*n)
        else:
            a[i] = ((2*(i+1))-1)/(4*(i+1))

        b[i] = 2 #Gerar vetor b

        c[i] = 1 - a[i] #Gerar vetor c
```

```
        d[i] = np.cos((2*np.pi*(i+1)*(i+1))/(n*n)) #Gerar vetor d

    v[0] = a[0] #Gerar vetor v
    v[-1] = c[n-2]

    return a,b,c,d,v #Retornar todos os vetores dessa[U+FFFD][U+FFFD]

def montarArray(x):
    y = np.zeros(len(x))
    n = len(x)

    for i in range(0, len(x)):
        if x[i] != "[" and x[i] != "]" and x[i] !=",":
            y[i] = int(x[i])

    return y, n

def montarMatrizVet():

    print("qual o vetor a? (modelo: [1,2,3]")
    a = str(input())
    aV, n = montarArray(a)

    print("qual o vetor b? (modelo: [1,2,3]")
    b = str(input())
    bV = montarArray(b)

    print("qual o vetor c? (modelo: [1,2,3]")
    c = str(input())
    cV = montarArray(c)

    print("qual o vetor d? (modelo: [1,2,3]")
    d = str(input())
    dV = montarArray(d)

    v = np.zeros(n-1)
    v[0] = a[0] #Gerar vetor v
    v[-1] = c[n-2]

    return aV, bV, cV, dV, v, n
```

A função seguinte, "tornarnaociclica"é responsável por retirar a última linha e coluna da matriz cíclica para transformar em tridiagonal e armazenar todos os vetores resultantes.

```
def tornarnaociclica(a,b,c,d,n):
    #Gerar vetores da matriz tridiagonal
    bT = np.copy(b[:n-1])
```

```
    aT = np.copy(a[:n-1])
    cT = np.copy(c[:n-1])

    aT[0] = 0
    cT[-1] = 0

    dT = np.copy(d[:n-1]) #Gerar vetor d_tio

    return aT, bT, cT, dT,
```

A função decompLU acha os coeficientes "l"e "u"das matrizes LU, resultantes da matriz A inicial decomposta em duas matrizes, uma diagonal superior e outra diagonal inferior, cujo produto resulta na A. Matematicamente descrito por: "$Ax = b -$"$(UL)x = b$"

```
def decompLU(a, b, c, n): #[U+FFFD][U+FFFD]respons[U+FFFD]vel por achar os
                                valores das matrizes L e U
    l = np.zeros(n)
    u = np.zeros(n)

    u[0] = b[0]
    for i in range(1,n) :
        if not (u[i-1] == 0):
            l[i] = a[i]/u[i-1] #Achar valores de L
        else:
            l[i] = 0

        u[i] = b[i] - l[i]*c[i-1] #Achar valores de U


    #print("A matriz L: " + str(l))
    #print("A matriz U: " + str(u))
    return l, u
```

Essa função, a partir das matrizes LU faz a solução dos sistemas descritos pelo método, o "Ly = b"e em seguida o "$Ux = y$", onde x é a solução do sistema "$Ax = b$"

```
def solucaoLU(l, u, c, d, n):
    y = np.zeros(n)

    #### SOLUCAO DO SISTEMA DE L
    y[0] = d[0]
    for i in range(1, n):
        y[i] = d[i]-l[i]*y[i-1]
    #print("A solucao do sistema L(y) obtida: " + str(y))

    #### SOLUCAO DO SISTEMA DE U
```

```
    x = np.zeros(n)
    x[n-1] = y[n-1]/u[n-1]
    for i in range(n-2, -1, -1):
        if not (u[i] == 0):
            x[i] = (y[i]-c[i]*x[i+1])/u[i]
        else:
            x[i] = 0
    #print("A solucao do sistema U(x) obtida: " + str(x))
    return x
```

Essas duas próximas funções são responsáveis por resolver de forma mais eficiente matrizes tridiagonais cíclicas utilizando o método da decomposição LU

```
def acharxT(zT, yT, a, b, c, d, n):
    xN = (d[n-1] - c[n-1]*yT[0] - a[n-1]*yT[n-2]) / (b[n-1] - c[n-1
                                        ]*zT[0] -a[n-1]*zT[n-2])
    xT = yT - xN*zT
    xT = np.append(xT, xN)

    return xT

def resolveciclica(n):
    #monta as matrizes
    a, b, c, d, v = montaMatrizN(n)
    print("Vetor A: " + str(a))
    print("Vetor B: " + str(b))
    print("Vetor C: " + str(c))
    aT, bT, cT, dT = tornarnaociclica(a, b, c, d, n)

    #faz a decomposicao
    l_c, u_c = decompLU(aT, bT, cT, n-1)
    y = solucaoLU(l_c, u_c, cT, dT, n-1)
    z = solucaoLU(l_c, u_c, cT, v, n-1)

    #encontra o valor final
    result = acharxT(z, y, a, b, c, d, n)

    return result
```