Victor Nascimento Pereira - 10773530

Gabriel Youssef Campos - 10884301

Métodos Numéricos e aplicações

Abril de 2022

Decomposição LU para matrizes Tridiagonais

O objetivo deste exercício programa é executar, de forma computacional, a decomposição de uma matriz A, n por n, $n > 1$, na forma LU, com L e U sendo ambas matrizes n por n. L seria Lower e U Uper, do inglês baixo e alto, pois são matrizes triangulares com zeros na parte de cima ou de baixo da diagonal principal, L com zeros na parte de cima e U com zeros na parte de baixo.

$$A = L\,x\,U$$

Método

Sendo A uma matriz triangularizável pelo Método da Eliminação de Gauss, sem trocas de linha, chamamos de U a matriz triangular superior obtida, e L a matriz triangular inferior formada pelos multiplicadores L_{ij}, i>j, com os elementos da diagonal principal iguais a 1.

Obtenção dos coeficientes de L e U:

Sabendo-se que A = LU, obtemos os coeficientes de L e U usando apenas as propriedades destas matrizes, pois, sendo L triangular inferior e U superior temos:

$$A_{ij} = \sum_{k=1}^{min\{i,j\}} L_{ik}U_{kj}$$

Por exemplo: temos L = $\begin{bmatrix} 1 & 0 & 0 \\ -0,5 & 1 & 0 \\ -0,25 & \frac{3}{8} & 1 \end{bmatrix}$ e U = $\begin{bmatrix} 8 & -4 & -2 \\ 0 & 8 & -3 \\ 0 & 0 & \frac{67}{8} \end{bmatrix}$

e utilizando a fórmula obtemos A = $\begin{bmatrix} 8 & -4 & -2 \\ -4 & 10 & -2 \\ -2 & -2 & 10 \end{bmatrix}$.

Usando que $L_{ii} = 1$ os coeficientes podem ser calculados da seguinte forma:

```
for i in range(N):
# Quando i=1 as somatórias (1) e (2) não são calculadas.
    if(i == 0):
        U[i, i:N] = A[i, i:N]
        L[(i+1):N, i] = (1/U[i, i])*(A[(i+1):N, i])
    elif(i != N-1):
        U[i, i:N] = A[i, i:N]-L[i, 0:i]@U[0:i, i:N]
        L[(i+1):N, i] = (1/U[i, i]) * (A[(i+1):N, i]
                                        -L[(i+1):N, 0:i]@U[0:i, i])
    # Quando i=N a expressão (2) não é calculada.
    else:
        U[i, i:N] = A[i, i:N]-L[i, 0:i]@U[0:i, i:N]
return U, L
```

O que nos leva a uma forma de resolução de sistemas lineares por um sistema triangular inferior e outro superior:

$$Ax = d \quad e \quad A = LU, \quad se\ tomarmos,\ Ly = d\ e\ Ux = y,\ temos\ LUx = d = Ax$$

A, L e U são matrizes nxn com n>1, e d, x e y são vetores de tamanho n.

Como L e U são triangulares, os sistemas Ly = d e Ux = y são também triangulares.

Matrizes tridiagonais

Matriz tridiagonal é uma matriz que, além dos elementos da diagonal principal, e das duas diagonais secundárias, que são a logo acima e a logo abaixo da principal, todos os elementos são nulos. Ou seja, se $|i - j| > 1,\ a_{ij} = 0$.

Podemos apresentar uma forma eficiente de obter L e U destas matrizes dada sua estrutura.

Se A é tridiagonal triangularizável pelo Método de Eliminação de Gauss sem trocas de linhas, os elementos que podem ser não nulos de U são U_{ii} e $U_{i,i+1}$,= $a_{i,i+1}$ e de L são $L_{i+1,i}$. O que resulta na aceleração da operação computacional, eliminando contas em que o resultado é sabidamente nulo.

Pensando computacionalmente, uma forma mais econômica para os recursos na máquina é armazenar a matriz tridiagonal em 3 vetores:

a = $(0, a_2, \ldots, a_{n-1}, a_n)$, b = $(b_1, b_2, \ldots, b_{n-1}, b_n)$, c = $(c_1, c_2, \ldots, c_{n-1},0)$

Sendo b a diagonal principal, a e b as diagonais inferior e superior, respectivamente.

O cálculo dos coeficientes $u_i = U_{ii}$ e $l_{i+1} = L_{i+1,i}$ da decomposição LU são feitos da seguinte forma:

```
    n=len(vetor_b) # n é a quantidade de elementos da diagonal principal

        # Construção da matriz A (tridiagonal) a partir de vetor_c,
vetor_b e vetor_a.
        A = np.zeros((n, n))

        for i in range(0, n): # Geração da diagonal principal
            A[i, i]=vetor_b[i]

        for i in range(0, n-1): # Geração das diagonais secundárias
            A[i,i+1]=vetor_c[i]
            A[i+1,i]=vetor_a[i+1]

        vetor_u = np.zeros(n)
        vetor_l = np.zeros(n)

        vetor_u[0]=vetor_b[0]

        for i in range(1, n):
            vetor_l[i]=vetor_a[i]/vetor_u[i-1]
            vetor_u[i]=vetor_b[i]-vetor_l[i]*vetor_c[i-1]
```

```
y = np.zeros(n)

x = np.zeros(n)


y[0]=d[0]

for i in range(1, n):

    y[i]=d[i]-vetor_l[i]*y[i-1]


x[n-1]=y[n-1]/vetor_u[n-1]


for i in range(n-2, -1, -1):

    x[i]=(y[i]-(vetor_c[i]*x[i+1]))/vetor_u[i]
```

Sistemas tridiagonais cíclicos

Uma matriz A tridiagonal ciclina é análoga à tridiagonal, porém, os elementos zerados de a e c, a_1 e c_n, são não nulos, posicionados da forma: a_1 em $A_{1,n}$, e c_n em $A_{n,1}$.

A tarefa pedida no exercício programa era que gerássemos nossos dados e resolvêssemos o sistema tridiagonal cíclico a partir da decomposição feita para um sistema tridiagonal.

A decomposição LU eficiente (com relação ao caso tridiagonall) desta matriz, ou seja, que não faz as contas que seriam sabidamente de resultado nulo, é da seguinte forma:

```
n=recebe_variaveis_usuario()

vetor_a=np.zeros(n)

vetor_c=np.zeros(n)

vetor_b=np.zeros(n)

vetor_d=np.zeros(n)


# Vetores definidos no ep:

# a=(0, a_2, ..., a_{n-1}, a_n)

# b=(b_1, b_2, ..., b_{n-1}, b_n)

# c=(c_1, c_2, ..., c_{n-1}, 0)


for i in range(n-1):

    vetor_a[i]=((2*(i+1))-1)/(4*(i+1))

    vetor_b[i]=2

    vetor_d[i]=math.cos((2*math.pi*(i+1)**2)/n**2)


vetor_a[n-1]=((2*n)-1)/(2*n) #a_n

vetor_b[n-1]=2


vetor_d[n-1]=math.cos(2*math.pi)


vetor_c=1-vetor_a


# Geração dos vetores v e w:
```

```
    vetor_v=np.zeros(n-1)
    vetor_w=np.zeros(n-1)
    vetor_v[0]=vetor_a[0]
    vetor_v[n-2]=vetor_c[n-2]
    vetor_w[0]=vetor_c[n-1]
    vetor_w[n-2]=vetor_a[n-1]

    l2, u2, y_til=decomposicao_lu_tridiagonal(vetor_c[0:n-1],
                           vetor_a[1:n], vetor_b[0:n-1], vetor_d[0:n-1])
    l3, u3, z_til=decomposicao_lu_tridiagonal(vetor_c[0:n-1],
                                  vetor_a[1:n], vetor_b[0:n-1], vetor_v)

    x_n=(vetor_d[n-1]-vetor_c[n-1]*y_til[0]-vetor_a[n-1]*y_til[n-2])/
            (vetor_b[n-1]-vetor_c[n-1]*z_til[0]-vetor_a[n-1]*z_til[n-2])

    x_til=y_til-(x_n*z_til)
```

CONCLUSÃO

Após realizarmos a implementação dos algoritmos, utilizamos a equação $\bar{x} = \bar{y} - x_n \bar{z}$ para verificar de o vetor $\bar{x}$ gerado estava de acordo com a equação; para vários valores diferentes de n, incluindo o valor de $n = 20$ pedido, pudemos chegar à conclusão de que a nossa implementação do algoritmo foi feita corretamente, com $\bar{x}$ de cada um dos métodos diferindo, no pior caso, de menos de $0,0001$ entre si.