Universidade de São Paulo

Instituto de Matemática e Estatística: Dep. Mat. Aplicada

Escola Politécnica

Luciano Chaparin Luisi - 9016866
luciano.luisi@usp.br
Bruno Prasinos Bernal - 10355141
brunobernal@usp.br

Decomposição LU para matrizes tridiagonais: EP1

Trabalho apresentado como avaliação da disciplina
MAP 3121 - Métodos Numéricos e Aplicações

São Paulo

2022

RESUMO

Neste trabalho foi estudada a implementação do algoritmo para a decomposição de uma matriz tridiagonal A n × n e também o algoritmo para a resolução de um sistema linear tridiagonal. As implementações foram feitas de forma a serem usadas como partes de outros programas e os algoritmos foram testados na resolução do sistema linear tridiagonal cíclico Ax = d, com coeficientes fornecidos.

1 INTRODUÇÃO

1.1 CONCEITOS

Em álgebra linear, a decomposição LU (em que *LU* vem do inglês *lower* e *upper*) é uma forma de fatoração de uma matriz não singular como o produto de uma matriz triangular inferior (*lower*) e uma matriz triangular superior (*upper*). Às vezes se deve pré-multiplicar a matriz a ser decomposta por uma matriz de permutação. Esta decomposição é utilizada em análise numérica para resolver sistemas de equações mais eficientemente ou encontrar as matrizes inversas.

1.2 OBJETIVOS

O objetivo do exercício programa consiste em implementar o algoritmo fornecido para a decomposição LU de uma matriz tridiagonal A n × n. As matrizes A, L e U deveriam ser armazenadas em vetores conforme descrito.

Também deveria ser implementado o algoritmo para a resolução de um sistema linear tridiagonal usando a decomposição LU da matriz.

As implementações deveriam ser feitas de forma que elas pudessem ser usadas como partes de outros programas e os algoritmos deveriam ser testados na resolução do sistema linear tridiagonal cíclico Ax = d, com os coeficientes da matriz fornecidos.

2 IMPLEMENTAÇÃO E TESTES

2.1 ARQUIVOS DO PROJETO

2.1.1 requirements

Diretório contendo arquivos listando os módulos e versões instalados no projeto, testado e executado em ambiente *Conda*.

2.1.2 testes_exemplos

Diretório que possui arquivos .CSV utilizados para os testes apresentados neste relatório (outros testes foram realizados com diferentes matrizes e dimensões, mas não anexados ao projeto).

2.1.3 LEIAME.txt

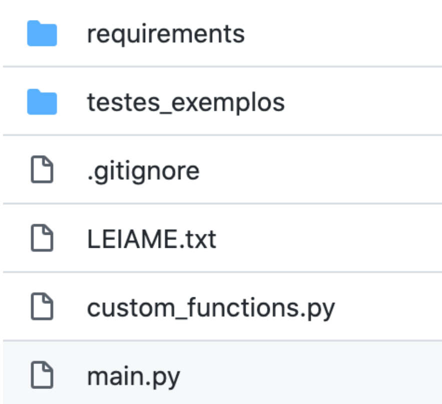
Documentação e instruções pertinentes ao uso do programa.

2.1.4 custom_functions.py

Módulo de funções usadas para recebimento, cálculo e devolução dos resultados, pertinentes ao exercício programa.

2.1.5 main.py

Executável que realiza o recebimento e retorno de dados do usuário num terminal através de linha de comando.



2.2 ALGORITMO

O programa foi escrito e testado em *Python 3.7*, utilizando as bibliotecas *NumPy* e *math* para a realização dos cálculos, e a biblioteca *tkinter* apenas para importação de arquivos .CSV.

2.2.1 Decomposição LU

A função recebe uma matriz quadrada como numpy array, de tamanho qualquer, cria duas listas vazias L e U para as matrizes inferior e superior respectivamente, substitui todos os elementos da diagonal de L por 1, e então itera para cada linha das matrizes a fim de calcular seus coeficientes, retornando uma tupla \``L, U`\`, de acordo com o seguinte procedimento simbólico:

para $i = 1, \cdots, n$ **faça**

$$U_{ij} = A_{ij} - \sum_{k=1}^{i-1} L_{ik} U_{kj}, \quad j = i, \cdots, n$$

$$L_{ji} = \left(A_{ji} - \sum_{k=1}^{i-1} L_{jk} U_{ki} \right) / U_{ii}, \quad j = i+1, \cdots, n$$

fim

```python
def decompLU(matriz: np.ndarray) -> 'tuple[np.ndarray, np.ndarray]':
    '''Recebe uma matriz e devolve sua decomposição LU···

    # Guarda o tamanho da matriz
    n = len(matriz)
    # Cria matriz 'L'(n por n) com zeros em todas as posições e transforma a lista em ndarray
    L = np.array([[0]*n]*n, float)
    # Itera para todas as linhas; todos os elementos da diagonal viram 1
    for i in range(0,n):
        L[i,i] = 1
    # Cria matriz 'U'(n por n) com zeros em todas as posições e transforma a lista em ndarray
    U = np.array([[0]*n]*n, float)
    # Itera para todas as linhas; realiza a decomposição (a exemplo das expressões 1 e 2 do enunciado)
    for i in range(0,n):
        U[i,i:] = matriz[i,i:] - np.dot(L[i,:i],U[:i,i:])
        L[(i+1):,i] = (1/U[i,i])*(matriz[(i+1):,i] - np.dot(L[(i+1):,:i],U[:i,i]))
    return L, U
```

2.2.2 Sistemas tridiagonais

O seguinte cálculo lança mão de duas funções; uma (\`*decompLUabc*\`) para decompor o sistema recebendo seus vetores diagonais, criando duas listas vazias \`*l*\` e \`*u*\` para os vetores diagonais das matrizes L e U, e calcula os coeficientes de cada um de acordo com o seguinte procedimento simbólico:

$$u_1 = b_1$$

para $i = 2, \cdots, n$ **faça**

$$l_i = a_i / u_{i-1} \text{ (multiplicador)}$$
$$u_i = b_i - l_i c_{i-1}$$

fim

E outra (`*solveLydUxy*`) que recebe os parâmetros devolvidos pela função anterior e resolve para um vetor `*d*`, retornando o vetor `*x*` de soluções, de acordo com o seguinte procedimento simbólico:

$Ly = d$:

$$y_1 = d_1$$

para $i = 2, \cdots, n$ **faça**

$$y_i = d_i - l_i y_{i-1}$$

fim

$Ux = y$:

$$x_n = y_n / u_n$$

para $i = n-1, \cdots, 1$ **faça**

$$x_i = (y_i - c_i\, x_{i+1}) / u_i$$

fim

```
def decompLUabc(a: np.ndarray, b: np.ndarray, c: np.ndarray) -> 'tuple[np.ndarray, np.ndarray]':
    '''Recebe 3 vetores diagonais de uma matriz e retorna sua decomposição LU (como vetores `l` e

    # Guarda o valor do vetor diagonal
    n = len(b)
    # Cria vetores com zero em todas as posições, de tamanhos 'n-1' e 'n'
    l = np.array([0]*(n-1), float)
    u = np.array([0]*n, float)
    # Primeiro termo do vetor 'u'
    u[0] = b[0]
    # Itera ao longo do tamanho do vetor diagonal; realiza a decomposição (a exemplo das expressõ
    for i in range(1,n):
        l[i-1] = a[i-1]/u[i-1]
        u[i] = b[i] - l[i-1]*c[i-1]
    return l, u
```

```
def solveLydUxy(l: np.ndarray, u: np.ndarray, c: np.ndarray, d: np.ndarray) -> np.ndarray:
    '''Resolve um sistema LUx=d a partir dos vetores subdiagonal da matriz inferior, diagonal e

    # Guarda o tamanho do sistema
    n = len(d)
    # Cria um vetor 'y' com zero em todas as posições
    y = np.array([0]*n, float)
    # Primeiro termo de 'y'
    y[0] = d[0]
    # Itera para o tamanho do sistema; expressão do exercício
    for i in range(1,n):
        y[i] = d[i] - l[i-1]*y[i-1]
    # Cria um vetor 'x' com zero em todas as posições
    x = np.array([0]*n, float)
    # Último termo de 'x'
    x[n-1] = y[n-1]/u[n-1]
    # Itera para o tamanho do sistema, do último para o primeiro termo; Expressão do exercício
    for i in range(n-2,-1,-1):
        x[i] = (y[i] - c[i]*x[i+1])/u[i]
    return x
```

2.2.3 Sistemas tridiagonais Cíclicos

A função recebe os vetores diagonais do sistema e o vetor de termos independentes, cria duas listas vazias \`v\` e \`w\` para receber os coeficientes calculados, e resolve o sistema de acordo com o seguinte procedimento simbólico:

$$\begin{gathered} T\tilde{x} + x_n v = \tilde{d} \\ w^t \tilde{x} + x_n b_n = d_n \end{gathered} \qquad x_n = \frac{d_n - c_n \tilde{y}_1 - a_n \tilde{y}_{n-1}}{b_n - c_n \tilde{z}_1 - a_n \tilde{z}_{n-1}} \quad e \quad \tilde{x} = \tilde{y} - x_n \tilde{z}$$

Em que T é a submatriz principal, resolvendo para $T\,\bar{y} = \bar{d}$ e $T\,\bar{z} = v$, retornando o vetor \`$x$\` de soluções.

```python
def solveCycTridi(a: np.ndarray, b: np.ndarray, c: np.ndarray, d: np.ndarray) -> np.ndarray:
    '''Resolve um sistema tridiagonal cíclico Ax = d a partir dos vetores diagonais e de term

    # Guarda o tamanho do vetor diagonal principal
    n = len(b)
    # Cria vetores vazios
    v = np.array([], float)
    w = np.array([], float)
    # Anexa o primeiro termo de 'a' ao vetor 'v' e o último termo de 'c' ao vetor 'w'
    v = np.append(v,a[0])
    w = np.append(w,c[n-1])
    # Itera para n-2 (dimSistema - 1)
    for t in range(1,n-2):
        v = np.append(v,0)
        w = np.append(w,0)
    # Anexa o penúltimo termo de 'c' ao vetor 'v' e o último termo de 'a' ao vetor 'w'
    v = np.append(v,c[n-2])
    w = np.append(w,a[n-1])
    # Resolve o sistema com a submatriz principal: Ty=d, Tz=v
    l, u = decompLUabc(np.delete(a,[0,n-1]),np.delete(b,n-1),np.delete(c,[n-1,n-2]))
    y = solveLydUxy(l,u,np.delete(c,[n-1,n-2]),np.delete(d,[n-1]))
    z = solveLydUxy(l,u,np.delete(c,[n-1,n-2]),v)
    # Encontra o último termo da solução X, em seguida os termos restantes
    x_n = (d[n-1]-c[n-1]*y[0]-a[n-1]*y[n-2])/(b[n-1]-c[n-1]*z[0]-a[n-1]*z[n-2])
    x = y - x_n*z
    x = np.append(x,x_n)
    x = np.array(x, float)
    return x
```

2.3 TESTES

A seguir são apresentados algumas capturas de tela do resultado de cada tipo de ação do programa, que apresenta ao usuário seus dados de entrada e resultados formatados. Os testes aqui apresentados não contemplam todos os testes, sendo apenas um demonstrativo simples do modelo de saída do programa.

2.3.1 Decomposição LU

DecompLU - Testes 1, 2 e 3

```
Decomposição A=LU
Matriz A:
[[ 1.  1.  0.  3.]
 [ 2.  1. -1.  1.]
 [ 3. -1. -1.  2.]
 [-1.  2.  3. -1.]]
Matriz inferior L:
[[ 1.  0.  0.  0.]
 [ 2.  1.  0.  0.]
 [ 3.  4.  1.  0.]
 [-1. -3.  0.  1.]]
Matriz superior U:
[[  1.   1.   0.   3.]
 [  0.  -1.  -1.  -5.]
 [  0.   0.   3.  13.]
 [  0.   0.   0. -13.]]
```

```
Decomposição A=LU
Matriz A:
[[ 8. -6.  2.]
 [-4. 11. -7.]
 [ 4. -7.  6.]]
Matriz inferior L:
[[ 1.   0.   0. ]
 [-0.5  1.   0. ]
 [ 0.5 -0.5  1. ]]
Matriz superior U:
[[ 8. -6.  2.]
 [ 0.  8. -6.]
 [ 0.  0.  2.]]
```

```
Decomposição A=LU
Matriz A:
[[ 1.  4.  1.]
 [ 1.  6. -1.]
 [ 2. -1.  2.]]
Matriz inferior L:
[[ 1.   0.   0. ]
 [ 1.   1.   0. ]
 [ 2.  -4.5  1. ]]
Matriz superior U:
[[ 1.  4.  1.]
 [ 0.  2. -2.]
 [ 0.  0. -9.]]
```

2.3.2 Sistemas tridiagonais

SisTridi - Teste 1

```
Sistema Ax=d
Matriz A:
[[ 2. -1.  0.  0.]
 [-1.  2. -1.  0.]
 [ 0. -1.  2. -1.]
 [ 0.  0. -1.  2.]]
Vetor d:
[1. 0. 0. 1.]
Solução X (valores encontrados):
[1. 1. 1. 1.]
Solução X (notação científica suprimida):
[1. 1. 1. 1.]
```

SisTridi - Teste 2

```
Sistema Ax=d
Matriz A:
[[ 2. -1.  0.  0.]
 [ 1.  2. -1.  0.]
 [ 0.  1.  2. -1.]
 [ 0.  0.  1.  2.]]
Vetor d:
[1. 0. 0. 1.]
Solução X (valores encontrados):
[ 0.44827586 -0.10344828  0.24137931  0.37931034]
Solução X (notação científica suprimida):
[ 0.44827586 -0.10344828  0.24137931  0.37931034]
```

2.3.3 Sistemas tridiagonais cíclicos

SisTridiCic - Teste 1

```
Sistema Ax=d
Matriz A:
[[ 2. -1.  0.  1.]
 [-1.  2. -1.  0.]
 [ 0. -1.  2. -1.]
 [ 1.  0. -1.  2.]]
Vetor d:
[0. 0. 0. 1.]
Solução X (valores encontrados):
[-0.5  0.   0.5  1. ]
Solução X (notação científica suprimida):
[-0.5  0.   0.5  1. ]
```

SisTridiCic - Teste 2

```
Sistema Ax=d
Matriz A:
[[ 2. -1.  0.  0.  0.  0.  0.  1.]
 [-1.  2. -1.  0.  0.  0.  0.  0.]
 [ 0. -1.  2. -1.  0.  0.  0.  0.]
 [ 0.  0. -1.  2. -1.  0.  0.  0.]
 [ 0.  0.  0. -1.  2. -1.  0.  0.]
 [ 0.  0.  0.  0. -1.  2. -1.  0.]
 [ 0.  0.  0.  0.  0. -1.  2. -1.]
 [ 1.  0.  0.  0.  0.  0. -1.  2.]]
Vetor d:
[0. 0. 0. 0. 0. 0. 0. 1.]
Solução X (valores encontrados):
[-1.5000000e+00 -1.0000000e+00 -5.0000000e-01  4.4408921e-16
  5.0000000e-01  1.0000000e+00  1.5000000e+00  2.0000000e+00]
Solução X (notação científica suprimida):
[-1.5 -1.  -0.5  0.   0.5  1.   1.5  2. ]
```

SisTridiCic - Teste 3

```
Sistema Ax=d
Matriz A:
[[ 2. -1.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  1.]
 [-1.  2. -1.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0. -1.  2. -1.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0. -1.  2. -1.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  2. -1.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0. -1.  2. -1.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0. -1.  2. -1.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0. -1.  2. -1.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0. -1.  2. -1.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0. -1.  2. -1.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0. -1.  2. -1.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0. -1.  2. -1.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0. -1.  2. -1.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0. -1.  2. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0. -1.  2. -1.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0. -1.  2. -1.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0. -1.  2. -1.]
 [ 1.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0. -1.  2.]]
Vetor d:
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
Solução X (valores encontrados):
[-4.00000000e+00 -3.50000000e+00 -3.00000000e+00 -2.50000000e+00
 -2.00000000e+00 -1.50000000e+00 -1.00000000e+00 -5.00000000e-01
  5.62050406e-16  5.00000000e-01  1.00000000e+00  1.50000000e+00
  2.00000000e+00  2.50000000e+00  3.00000000e+00  3.50000000e+00
  4.00000000e+00  4.50000000e+00]
Solução X (notação científica suprimida):
[-4.  -3.5 -3.  -2.5 -2.  -1.5 -1.  -0.5  0.   0.5  1.   1.5  2.   2.5
  3.   3.5  4.   4.5]
```

SisTridiCic - Teste 4, padrão do ep, n=5

```
Sistema Ax=d
Matriz A:
[[2.         0.75       0.         0.         0.25      ]
 [0.375      2.         0.625      0.         0.        ]
 [0.         0.41666667 2.         0.58333333 0.        ]
 [0.         0.         0.4375     2.         0.5625    ]
 [0.1        0.         0.         0.9        2.        ]]
Vetor d:
[ 0.96858316  0.53582679 -0.63742399 -0.63742399  1.        ]
Solução X (valores encontrados):
[ 0.2888525   0.29033825 -0.24507104 -0.45986772  0.69249785]
Solução X (notação científica suprimida):
[ 0.2888525   0.29033825 -0.24507104 -0.45986772  0.69249785]
```

SisTridiCic - Teste 5, padrão do ep, n=20

A=

```
[[2.         0.75       0.         0.         0.         0.         0.         0.         0.         0.
0.         0.     0.         0.         0.         0.         0.         0.     0.         0.25       ]
 [0.375      2.         0.625      0.         0.         0.         0.         0.         0.         0.
0.         0.     0.         0.         0.         0.         0.         0.     0.         0.         ]
 [0.         0.41666667 2.         0.58333333 0.         0.         0.         0.         0.         0.
0.         0.     0.         0.         0.         0.         0.         0.     0.         0.         ]
 [0.         0.         0.4375     2.         0.5625     0.         0.         0.         0.         0.
0.         0.     0.         0.         0.         0.         0.         0.     0.         0.         ]
 [0.         0.         0.         0.45       2.         0.55       0.         0.         0.         0.
0.         0.     0.         0.         0.         0.         0.         0.     0.         0.         ]
 [0.         0.         0.         0.         0.45833333 2.         0.54166667 0.         0.         0.
0.         0.     0.         0.         0.         0.         0.         0.     0.         0.         ]
 [0.         0.         0.         0.         0.         0.46428    2.         0.53571429 0.         0.
0.         0.     0.         0.         0.         0.         0.         0.     0.         0.         ]
 [0.         0.         0.         0.         0.         0.         0.46875    2.         0.53125    0.
0.         0.     0.         0.         0.         0.         0.         0.     0.         0.         ]
 [0.         0.         0.         0.         0.         0.         0.         0.47222222 2.         0.52777778
0.         0.     0.         0.         0.         0.         0.         0.     0.         0.         ]
 [0.         0.         0.         0.         0.         0.         0.         0.         0.475      2.
0.525      0.     0.         0.         0.         0.         0.         0.     0.         0.         ]
 [0.         0.         0.         0.         0.         0.         0.         0.         0.         0.47727273
2.         0.52272  0.         0.         0.         0.         0.         0.     0.         0.         ]
 [0.         0.         0.         0.         0.         0.         0.         0.         0.         0.
0.47916667 2.     0.52083333 0.         0.         0.         0.         0.     0.         0.         ]
 [0.         0.         0.         0.         0.         0.         0.         0.         0.         0.
0.         0.480769 2.         0.51923077 0.         0.         0.         0.     0.         0.         ]
 [0.         0.         0.         0.         0.         0.         0.         0.         0.         0.
0.         0.     0.48214286 2.         0.51785714 0.         0.         0.     0.         0.         ]
 [0.         0.         0.         0.         0.         0.         0.         0.         0.         0.
0.         0.     0.         0.48333333 2.         0.51666667 0.         0.     0.         0.         ]
 [0.         0.         0.         0.         0.         0.         0.         0.         0.         0.
0.         0.     0.         0.         0.484375   2.         0.515625   0.     0.         0.         ]
 [0.         0.         0.         0.         0.         0.         0.         0.         0.         0.
0.         0.     0.         0.         0.         0.48529412 2.         0.514705 0.         0.         ]
 [0.         0.         0.         0.         0.         0.         0.         0.         0.         0.
0.         0.     0.         0.         0.         0.         0.48611111 2.     0.51388889 0.         ]
 [0.         0.         0.         0.         0.         0.         0.         0.         0.         0.
0.         0.     0.         0.         0.         0.         0.         0.486842 2.         0.51315789]
 [0.025      0.         0.         0.         0.         0.         0.         0.         0.         0.
0.         0.     0.         0.         0.         0.         0.         0.     0.975      2.         ]]
```

d = [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 1.00000000e+00]

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]

4 RESULTADOS

Sucessos e limitações observadas

Testes para decomposição LU, sistemas tridiagonais e tridiagonais cíclicos apresentaram resultados consistentes para matrizes de qualquer tamanho, e compatíveis com resoluções manuais calculadas para sistemas pequenos ($n<5$). O algoritmo não verifica se uma matriz recebida para sistemas tridiagonais (cíclicos ou não) é de fato uma matriz tridiagonal. A devolução dos resultados no terminal pode perder a formatação para boa visualização do usuário para sistemas muito grandes. O vetor de termos independentes \`$d$\`e o de resultados \`$x$\` na resolução de sistema tridiagonal cíclico para sistemas muito extensos no modelo do exercício programa passam a apresentar valores com muitas casas decimais por conta da limitação dos coeficientes de \`$d$\`dentro da função cosseno ($-1 < d_i < 1, \forall i$), imprimindo também os valores com notação científica suprimida para visualização dos dados.