Relatório EP1

Aluno: Rafael Yamada de Oliveira
número USP: 11803890

O principal tema do EP 1 foi a decomposição LU de matrizes e suas aplicações e, nesse sentido, criar um algoritmo em C/C++ ou python capaz de decompor certas matrizes em LU e, dessa forma, resolver sistemas lineares. O principal tipo de matriz discutido foram as tridiagonais que são matrizes quadradas cujos os únicos elementos da diagonal principal e as que estão acima e abaixo a ela são não nulas.

O EP foi dividido em três principais partes: Decomposição LU de matrizes triangularizáveis pelo Método de Eliminação de Gauss sem trocas de linhas e sem a necessidade de condensação pivotal para a estabilidade numérica. Decomposição LU de matrizes tridiagonais. E, resolução de um sistema linear tridiagonal usando a decomposição LU da matriz tridiagonal.

A implementação dos algoritmos foi feita em Python 3.10.4 e com o uso das bibliotecas numpy e math e com o auxílio da IDE VScode.

Na primeira parte do EP, o algoritmo recebe uma matriz quadrada M que seja triangularizáveis pelo Método de Eliminação de Gauss sem trocas de linhas e sem a necessidade de condensação pivotal para a estabilidade numérica e seu tamanho n. No algoritmo são criadas duas arrays: uma matriz quadrada U n × n preenchida com zeros e uma matriz identidade L de tamanho n. Os arrays U e L tem seus valores atualizados de forma a decompor a matriz M em LU pelo seguinte algoritmo:

```python
for i in range(0,n):
    U[0][i] = M[0][i]

for i in range(1,n):
    L[i][0] = M[i][0]/U[0][0]


for i in range(1,n):
    U[i, i : n] = M[i, i : n] - np.dot(L[i, 0 : (i)],U[0:i, i : n])
    if i < n-1:
        L[(i + 1) : n, i] =  1/U[i,i] * (M[(i + 1) : n, i] - np.dot(L[(i + 1) : n, 0: i] , U[0 : i, i]))
```

1.1

Os testes podem ser feitos por uma interface no prompt onde o usuário digita os valores da matriz M e são devolvidos os valores de M, L e U no prompt. Segue exemplo da matriz M =

$$\begin{bmatrix} 8 & -4 & -2 \\ -4 & 10 & -2 \\ -2 & -2 & 10 \end{bmatrix}$$

```
C:\Users\rafay\OneDrive\Ambiente de Trabalho\EP1\dist\EP11.exe
------------ OPCAO 1 ------------

Digite o tamanho da matriz: 3
Elemento da linha 1 , coluna 1 : 8
Elemento da linha 1 , coluna 2 : -4
Elemento da linha 1 , coluna 3 : -2
Elemento da linha 2 , coluna 1 : -4
Elemento da linha 2 , coluna 2 : 10
Elemento da linha 2 , coluna 3 : -2
Elemento da linha 3 , coluna 1 : -2
Elemento da linha 3 , coluna 2 : -2
Elemento da linha 3 , coluna 3 : 10
A decomposicao da matriz M  =
 [[ 8. -4. -2.]
 [-4. 10. -2.]
 [-2. -2. 10.]]
U =
 [[ 8.    -4.    -2.   ]
 [ 0.     8.    -3.   ]
 [ 0.     0.     8.375]]
L =
 [[ 1.     0.     0.   ]
 [-0.5    1.     0.   ]
 [-0.25  -0.375  1.   ]]
```

2.1

Para segunda parte do EP (Decompor uma matriz tridiagonal em LU),a função “decomposicaotridiagonalLU” é implementada, onde seu algoritmo recebe o tamanho e os vetores das três diagonais ($a1$, $b1$, $c1$) sendo o $b1$ a diagonal principal, $a1$ a diagonal secundária inferior e $c1$ a diagonal secundária superior. No algoritmo são criados dois vetores uv e lv que recebem os valores da decomposição LU da matriz tridiagonal cujos vetores são $a1$, $b1$, $c1$. Nesse sentido, vetores uv e lv são retornados pela função. Segue o algoritmo de decomposição LU:

```
uv[0]= b1[0]

for i in range(1, b1.size):
    lv[i] = a1[i]/uv[i-1]
    uv[i] = b1[i] - (lv[i]*c1[i-1])
```

1.2

Os testes podem ser feitos por uma interface no prompt onde o usuário digita os valores dos vetores $a1$, $b1$, $c1$ e a interface devolve os valores dos vetores uv e lv.

Segue exemplo com os vetores $a1$, $b1$, $c1$ abaixo

$a1 \ = \ (0, 1, 1, 1)$

$b1 \ = \ (-\ 2, -\ 2\ , -\ 2\ , -\ 2)$

$c1 \ = \ (1, 1, 1, 0)$

```
C:\Users\rafay\OneDrive\Ambiente de Trabalho\EP1\dist\EP11.exe
------------ OPCAO 2 ------------

Digite o tamanho da matriz: 4
digite 1o. valor de a 0
digite 2o. valor de a 1
digite 3o. valor de a 1
digite 4o. valor de a 1
digite 1o. valor de b -2
digite 2o. valor de b -2
digite 3o. valor de b -2
digite 4o. valor de b -2
digite 1o. valor de c 1
digite 2o. valor de c 1
digite 3o. valor de c 1
digite 4o. valor de c 0
decomposicao LU da matriz tridiagona M =

L =
 [ 0.         -0.5        -0.66666667 -0.75      ]
U =
 [-2.         -1.5        -1.33333333 -1.25      ]
```

2.2

Na terceira parte do EP (resolução de um sistema linear tridiagonal usando a decomposição LU da matriz tridiagonal) é dividida em dois casos: No caso do sistemas cíclicos e não cíclicos. Para identificar os casos basta observar se existe um valor diferente de 0 nas posições 1 do vetor a e na posição n no vetor c. Como pode ser conferido no algoritmo abaixo.

```
if a1[0] == 0 or c1[c1.size - 1] == 0:
    return  0
else:
    return 1
```

1.3

Caso o sistema seja não cíclico, primeiramente decompõe-se a matriz em LU chamando a função “decomposicaotridiagonalLU”. Para resolução do sistema linear a função “SolucaoSL” é implementada, seu algoritmo recebe os vetores l, u, d, p, n, onde l é o vetor lv e u é o vetor uv ambos devolvidos pela função “decomposicaotridiagonalLU”. d é o vetor com os resultados da igualdade do sistema linear $Ax = d$. p é o vetor $c1$ e n é o tamanho do vetor. Para a resolução do sistema linear é criado um vetor auxiliar y de dimensão n. Nesse sentido, para encontrar a solução do sistema linear é implementado o algoritmo abaixo onde x é a solução do sistema linear:

```
y[0] = d[0]

for i in range(1, n):
    y[i] = d[i] - l[i]*y[i-1]

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

for i in range(0, n-1):
    x[n - 2 - i] = (y[n - 2 - i] - p[n - 2 - i]*x[n - 1 - i])/u[n - 2 - i]
```

1.4

Os testes podem ser feitos por uma interface no prompt onde o usuário digita os valores dos vetores $a1$, $b1$, $c1$ e a interface devolve o valor de x e ainda informa se o sistema é ou não cíclico. Segue exemplo com

$a1 = (0, 1, 1, 1)$

$b1 = (-2, -2, -2, -2)$

$c1 = (1, 1, 1, 0)$

```
C:\Users\rafay\OneDrive\Ambiente de Trabalho\EP1\dist\EP11.exe
----------- OPCAO 3 -----------

Digite o tamanho da matriz: 4
digite 1o. valor de a 0
digite 2o. valor de a 1
digite 3o. valor de a 1
digite 4o. valor de a 1
digite 1o. valor de b -2
digite 2o. valor de b -2
digite 3o. valor de b -2
digite 4o. valor de b -2
digite 1o. valor de c 1
digite 2o. valor de c 1
digite 3o. valor de c 1
digite 4o. valor de c 0
digite 1o. valor de d 5
digite 2o. valor de d 1
digite 3o. valor de d 0
digite 4o. valor de d 8

Sistema tridiagonal nao eh ciclico
solucao =  [-6.2 -7.4 -7.6 -7.8]
```

2.3

Para o caso cíclico do sistema linear $Mx = d$, reduz-se o problema para um caso não cíclico criando uma submatriz $T = M(n - 1) \times (n - 1)$ cíclica e posteriormente resolve-se o sistema por completo. Nesse sentido, primeiramente, decompõe-se a matriz em LU chamando a função "decomposicaotridiagonalLU" porém, agora, na função são criadas 5 vetores auxiliares $(lc,\ uc,\ cc,\ v,\ e\ w)$ $lc,\ uc,\ cc$ que recebem, basicamente, os mesmos valores valores $lv,\ uv,\ e\ c1$ porém, descartando o último valor dos vetores $a1,\ b1,\ c1$. Segue seu algoritmo (note que esse algoritmo é muito similar ao algoritmo 1.2 mas agora, o laço for vai de 1 a b1.size - 1).

```
uc[0]= b1[0]

for i in range(1, b1.size - 1):
    lc[i] = a1[i]/uc[i-1]
    uc[i] = b1[i] - (lc[i]*c1[i-1])
```

1.5

Posteriormente são implementados os vetores v e w onde $v = (a1,\ 0,\ .\ .\ .\ ,\ 0,\ c_{n-1})$ e $w = (c_n,\ 0,\ .\ .\ .\ ,\ 0,\ a_n)$ por meio do seguinte algoritmo:

```
v[0] = a1[0]
v[v.size - 1] = c1[c1.size - 2]
w[0] = c1[c1.size - 1]
w[w.size - 1] = a1[a1.size - 1]
```

1.6

Para a resolução do sistema linear, primeiramente resolve-se o sistema cíclico $Ty' = d'$,note que $d' = (d_1,\ .\ .\ .\ ,\ d_{n-1})$, e $Tz' = v$, note que $v = (a1,\ 0,\ .\ .\ .\ ,\ 0,\ c_{n-1})$ usando o algoritmo de "SolucaoSL" que retornará respectivamente o valor de y' e z'. tendo esse vetores, é possível calcular x_n por meio da equação abaixo:

$$x_n = \frac{d_n - c_n \tilde{y}_1 - a_n \tilde{y}_{n-1}}{b_n - c_n \tilde{z}_1 - a_n \tilde{z}_{n-1}}$$

Cujo código do algoritmo é:

```
x = (d[d.size - 1] - ( w[0] * y[0] ) - (w[w.size - 1] * y[y.size - 1]))/(b[b.size - 1] - (w[0] * z[0]) - (w[w.size - 1]  * z[z.size - 1]))
return x
```

1.7

Com o valor de x_n é possível calcular a solução x completa por meio da equação $x' = y' - x_n z'$ note que $x = (x_1, ..., x_{n-1})$. Segue código algoritmo dessa equação:

```
xx = np.zeros(n)
xx[n - 1] = xn
x = y - np.dot(xn, z)

for i in range (n - 1):
    xx[i] = x[i]

return xx
```

1.8

Os testes podem ser feitos por uma interface no prompt onde o usuário digita os valores dos vetores $a1$, $b1$, $c1$ e a interface devolve o valor de x e ainda informa se o sistema é ou não cíclico. Segue exemplo com

$a1 = (2, 1, 1, 1)$

$b1 = (-2, -2, -2, -2)$

$c1 = (1, 1, 1, 2)$

```
C:\Users\rafay\OneDrive\Ambiente de Trabalho\EP1\dist\EP11.exe
------------ OPCAO 3 ------------

Digite o tamanho da matriz: 4
digite 1o. valor de a 2
digite 2o. valor de a 1
digite 3o. valor de a 1
digite 4o. valor de a 1
digite 1o. valor de b -2
digite 2o. valor de b -2
digite 3o. valor de b -2
digite 4o. valor de b -2
digite 1o. valor de c 1
digite 2o. valor de c 1
digite 3o. valor de c 1
digite 4o. valor de c 2
digite 1o. valor de d 5
digite 2o. valor de d 1
digite 3o. valor de d 0
digite 4o. valor de d 8
sistema tridiagonal eh ciclico
solucao =  [7.36363636 6.45454545 6.54545455 6.63636364]
```

2.4

No enunciado é do EP sugerido é sugerido um teste do algoritmo para um sistema linear cíclico $Ax = d$ onde os coeficientes da matriz são:

$$a_i = \frac{2i-1}{4i},\ 1 \leq i \leq n-1,\ a_n = \frac{2n-1}{2n},$$
$$c_i = 1 - a_i,\ 1 \leq i \leq n,$$
$$b_i = 2,\ 1 \leq i \leq n,$$

e o lado direito "d" do sistema linear é:

$$d_i = \cos\left(\frac{2\pi i^2}{n^2}\right),\ 1 \leq i \leq n.$$

Nesse sentido esse teste foi implementado com o seguinte algoritmo:

```
def testeA (n):

    at = np.zeros(n)

    for i in range (0, n - 1):
        at[i] = (2 * (i + 1) - 1 )/(4* (i + 1))

    at[n - 1] = (2 * n - 1)/(2 * n)

    return at

def testeB (n):

    bt = np.zeros(n)

    for i in range (0, n):
        bt[i] = 2

    return bt

def testeC (n, ai):
    ct = np.zeros(n)

    for i in range (0, n):
        ct[i] = 1 - ai[i]

    return ct

def testeD (n):
    d = np.zeros(n)

    for i in range(0,n):
        d[i] = math.cos(2 * math.pi * (i + 1)**2/(n**2))

    return d
```

1.9

Onde os vetores a, b, c, d são os retornos das funções testeA, testeB, testeC e testeD respectivamente. E n é o tamanho do vetor. O teste pode ser feito por meio da interface no prompt onde o usuário digita o valor de n e, após executar o algoritmo do teste, é mostrado na tela os vetores de a, b, c, d e o vetor com as soluções do sistema linear. Segue exemplo para n = 20:

```
C:\Users\rafay\OneDrive\Ambiente de Trabalho\EP1\dist\EP11.exe
------------ OPCAO 4 ------------

Digite o tamanho da matriz (sugestao do enunciado n = 20): 20
Vetor A =  [0.25       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 0.975     ]
Vetor B =  [2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2.]
Vetor C =  [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
 0.51315789 0.025     ]
Vetor 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]
solucao =  [ 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]
```