**UNIVERSIDADE DE SÃO PAULO**

**ESCOLA POLITÉCNICA**


MAP3121 - Métodos Numéricos e Aplicações

# Relatório - EP1

Professor: Henrique Von Dreifus

Tiago Simões Inácio Cavalcanti n° USP: 11804891

Pedro Maia da Silva n° USP: 11805996

São Paulo-SP,

maio de 2022

## Introdução

O EP proposto pela disciplina de Métodos Numéricos tem como objetivo utilizar o método da decomposição LU para resolver problemas envolvendo matrizes tridiagonais e matrizes tridiagonais cíclicas. Além disso, no enunciado é apresentado um problema de uma matriz 20x20, o qual deverá ser solucionado e as respostas mostradas neste relatório. Vale ressaltar que o nosso código está bastante comentado, então no relatório não terão comentários minuciosos sobre as funções.

## Metodologia

Separou-se o EP em quatro partes diferentes. sendo elas: o algoritmo da decomposição LU para qualquer matriz, o algoritmo da decomposição LU para matrizes tridiagonais, algoritmo LU para matrizes tridiagonais cíclicas e, por fim, rodada a tarefa dada no enunciado e discutir os seus resultados.

## Decomposição LU

LU é um método utilizado em análise numérica que consiste em decompor uma matriz A em uma matriz triangular inferior (L) e uma matriz triangular superior (U), a fim de resolver sistemas de equações lineares com maior eficiência. Dessa forma, nós escrevemos um código para essa parte descrita no enunciado, utilizando as equações mostradas e implementando-as no phyton, utilizando cinco funções:

a) def MatrizIdentidade(n): serve para gerar uma matriz identidade, já que a diagonal principal da matriz L é 1. Assim, apenas os valores abaixo dessa diagonal serão preenchidos.
b) def DecomposiçãoLU(A): realiza a decomposição da matriz A, retornando L e a matriz A novamente, já modificada, que equivale a U.
c) def Sub_Retroativa(A , b): função que resolve a equação Ux = b, através de substituições retroativas, ou seja, que partem de cima para baixo, já que U é triangular superior.
d) def Sub_Sucessiva(A , b): resolve a equação Lx = b, através de substituições sucessivas, ou seja, que partem de baixo para cima, já que L é triangular inferior.
e) def ResoluçãoSistema(L, U, b): função que aplica os dois algoritmos de substituição retroativa e sucessiva para resolver o sistema linear Ly=b e Ux=y, devolvendo o vetor x.

Vale relembrar que as entradas do código são a matriz A (n x n) e a matriz b (n x 1), e o intuito do código é resolver Ax=b pelo método da decomposição LU.

**Matrizes Tridiagonais**

As matrizes tridiagonais são aquelas que possuem a diagonal principal, a diagonal logo acima dela e a diagonal logo abaixo dela não nulas, enquanto todos os outros elementos não nulos. Assim, pensando em uma matriz 4x4 tridiagonal, por exemplo:

| | | | |
|---|---|---|---|
| b1 | c1 | 0 | 0 |
| a2 | b2 | c2 | 0 |
| 0 | a3 | b3 | c3 |
| 0 | 0 | a4 | b4 |

Percebe-se que podemos fazer uma adaptação na decomposição LU para esse caso, já que o nosso algoritmo anterior utiliza todos os elementos da matriz para calcular L e U. Nesse caso, não há a necessidade de utilizar elementos para calcular L e U, apenas as três diagonais. O computador só estaria perdendo tempo calculando várias equações que resultam obrigatoriamente em 0. Por exemplo, uma matriz 20x20, utilizamos somente 58 elementos (as três diagonais) em vez de 400 elementos, otimizando muito o processo.

Sobre as matrizes L e U, a matriz L possuirá a diagonal principal valendo 1 em todos os seus elementos, e a diagonal abaixo dela (Li) não nula, os outros elementos são obrigatoriamente 0. A matriz U terá a diagonal principal (Uii) e a diagonal acima dela (Ui) também não nula, os outros elementos são obrigatoriamente nulos.

Assim, utilizando o que foi exposto no enunciado, assim como as lógicas já fornecidas, foi possível desenvolver um algoritmo que recebe três vetores da matriz A, o vetor “a” referente a diagonal abaixo da principal, o vetor “b” referente a diagonal principal e o vetor “c” referente a diagonal acima da principal.

O algoritmo devolve Li, Ui e Uii (explicados anteriormente), além de resolver os sistemas Ly=d e Ux=y. As funções utilizadas foram:

a) def GeradorDaMatriz(n); serve para gerar os vetores a, b e c de acordo com o que foi estabelecido no enunciado.
b) def GeradorLadoDireito(n); serve para gerar o vetor d de acordo com o que foi estabelecido no enunciado.
c) def LUTridiagonal(n): função que recebe os três vetores (a,b,c) e devolve Li,Uii,Ui
d) def SistemaTridiagonal(li, ui, uii, d): função que recebe li, ui, uii, d e devolve o vetor solução x.

**Matriz tridiagonal cíclica**

A matriz tridiagonal cíclica tem uma peculiaridade, os elementos mais distantes das diagonais, ou seja, as extremidades, ou seja, o elemento da linha 1 e coluna n e da linha n e coluna 1 são não nulos. Assim, para resolver esse tipo de matriz, precisamos de um tratamento diferente. Se considerarmos apenas a submatriz (T) com dimensões (n-1)x(n-1), teremos uma matriz tridiagonal, dessa forma, podemos utilizar tudo que já foi feito anteriormente. Logo, calcula-se Ty = d e Tz = v, sendo v explicitado no enunciado .
Com y e z calculados, utiliza-se a fórmula dada para calcular Xn. Após encontrar o valor de Xn, utiliza-se a fórmula de x dada para encontrar os outros valores desse vetor.

**Tarefa**

Com o que foi oferecido na tarefa, observa-se que:

```
Vetor a: [0.25, 0.375, 0.4166666666666667, 0.4375, 0.45,
0.4583333333333333, 0.4642857142857143, 0.46875, 0.4722222222222222,
0.475, 0.4772727272727273, 0.4791666666666667, 0.4807692307692308,
0.48214285714285715, 0.48333333333333334, 0.484375, 0.4852941176470588,
0.4861111111111111, 0.4868421052631579, 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, 0.5833333333333333, 0.5625, 0.55,
0.5416666666666667, 0.5357142857142857, 0.53125, 0.5277777777777778,
0.525, 0.5227272727272727, 0.5208333333333333, 0.5192307692307692,
0.5178571428571428, 0.5166666666666666, 0.515625, 0.5147058823529411,
0.5138888888888888, 0.513157894736842, 0.025000000000000022]

Vetor d; [0.9998766324816606, 0.9980267284282716, 0.9900236577165575,
0.9685831611286311, 0.9238795325112867, 0.8443279255020151,
0.7181262977631888, 0.5358267949789965, 0.2940403252323041,
6.123233995736766e-17, -0.32391741819814945, -0.6374239897486897,
-0.8837656300886935, -0.9980267284282716, -0.9238795325112868,
-0.6374239897486895, -0.17192910027940964, 0.3681245526846774,
0.8181497174250233, 1.0]

Vetor Li: [0.125, 0.1875, 0.20865936358894108, 0.23529411764705882,
0.24069475446428573, 0.24605263157894736, 0.24899893990272545,
0.25138482164908443, 0.2531852164537873, 0.2546466909810157,
0.2558423564609984, 0.25684276446373655, 0.2576907716254809,
0.2584189253566812, 0.2590507800525869, 0.2596042251107881,
0.2600929629600346, 0.2605276956114953, 0.26091689140728686,
0.5225346798582311]
```

**Vetor Ui: [2, 1.996875, 1.859375, 1.8695878977569118, 1.8627450980392157, 1.8646092006138393, 1.864671052631579, 1.865125574219357, 1.8653295598308477, 1.8654953537589254, 1.865603135315575, 1.8656827628579757, 1.8657412822121378, 1.8657860564450621, 1.8658209426032617, 1.8658487031870532, 1.865871150359426, 1.8658895659737322, 1.8659048625529069, 1.8659177085823666]**

**Vetor Uii: [0.75, 0.625, 0.5833333333333333, 0.5625, 0.55, 0.5416666666666667, 0.5357142857142857, 0.53125, 0.5277777777777778, 0.525, 0.5227272727272727, 0.5208333333333333, 0.5192307692307692, 0.5178571428571428, 0.5166666666666666, 0.515625, 0.5147058823529411, 0.5138888888888888, 0.513157894736842, 0.025000000000000022]**

**Vetor solução X: [0.2446000686207052, 0.050039918554712, -0.1441886865670005, -0.2871107812137882, -0.3545258176386563, -0.3468161894918134, -0.28057659677109925, -0.17919233031424264, -0.06528071279377118, 0.043991879108510774, 0.13807091524303441, 0.21129580016232807, 0.26217376407805626, 0.29689135292043983, 0.3214593495506124, 0.3345736438220244, 0.31149950124040127, 0.33511128228711806, 0.32703433730844056, 0.3375132597043764]**