Sistemas Lineares Triangulares

Decomposição LU

A **Decomposição LU** é a fatoração de uma matriz quadrada A em duas matrizes: L (triangular inferior) e U (triangular superior), de forma que $A = LU$.

- L é uma matriz triangular inferior com **elementos diagonais iguais a 1**.
- U é uma matriz triangular superior.

A decomposição LU **facilita a resolução de sistemas lineares**, especialmente quando o sistema é a parte de múltiplos sistemas com a mesma matriz de coeficientes.

Foi desenvolvida a partir do método de **Eliminação de Gauss**, adaptando-o para uma fatoração matricial.

https://en.wikipedia.org/wiki/LU_decomposition
https://en.wikipedia.org/wiki/Gaussian_elimination

Lower Triangular

$$\begin{bmatrix} A00 & A01 & A02 \\ A10 & A11 & A12 \\ A20 & A21 & A22 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ L10 & 1 & 0 \\ L20 & L21 & 1 \end{bmatrix} \begin{bmatrix} U00 & U01 & U02 \\ 0 & U11 & U12 \\ 0 & 0 & U22 \end{bmatrix}$$

Upper Triangular

Exercício

Calcule a decomposição **LU** da matriz A abaixo.

$$A = \begin{bmatrix} 1 & 4 & -3 \\ -2 & 8 & 5 \\ 3 & 4 & 7 \end{bmatrix}$$

Condições para a Decomposição LU

Existência:

A matriz A deve ser **não singular**.

Eventualmente, pode ser necessário alterar a ordem das linhas e/ou colunas de A para evitar pivôs nulos (**pivoteamento**).

$$PA = LU,$$

onde P é uma matriz de permutação.

Não Singularidade
A matriz A deve ter determinante diferente de zero.

Sistemas Equivalentes
Dois sistemas lineares são equivalentes se possuem a mesma solução.

Pivoteamento Parcial
Rearranjo das linhas (ou colunas) para evitar pivôs nulos.

Strang, Gilbert. *Linear Algebra and Its Applications*. Cengage Learning, 2016.

Exercício

Determine a decomposição **LU** da matriz A abaixo.

$$A = \begin{bmatrix} 0 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

Decomposição LU

Teorema:

Dada uma matriz quadrada A que pode ser fatorada como $A = LU$, onde L é triangular inferior com elementos diagonais iguais a 1 e U é triangular superior, então tal fatoração é única.

Conexão com Eliminação de Gauss:

A decomposição **LU** é essencialmente a **Eliminação de Gauss** aplicada de forma a registrar as operações de eliminação em uma matriz triangular inferior L.

Operações Elementares vs Fator L

Na eliminação de Gauss, as operações elementares ão aplicadas a uma matriz A para transformá-la em uma matriz triangular superior U.

$$E_n \cdots E_2 E_1 A = U$$

Portanto,

$$A = (E_n \cdots E_2 E_1)^{-1} U = E^{-1} U$$

- As matrizes elementares são triangulares inferiores.
- O produto $E = E_n \cdots E_2 E_1$ é triangular inferior.
- A inversa de uma matriz triangular inferior é triangular inferior.
- Portanto, E^{-1} é triangular inferior e igual a L.

Algoritmo de Decomposição LU

1. **Inicialize** L como uma matriz identidade e U como uma cópia de A.
2. **Para** cada coluna j de A:
   - **Para** cada linha i abaixo da diagonal ($i > j$):
     - Calcule o fator de eliminação: $L_{ij} = U_{ij}/U_{jj}$.
     - **Subtraia** $L_{ij} \times U_{jk}$ de U_{ik} para todas as colunas k a partir de j.
3. **Repita** até que U seja uma matriz triangular superior.

Demonstração da Decomposição LU:

1. Passo Inicial:

$$L = \begin{bmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{bmatrix}, \quad U = \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix}$$

2. Aplicação da Eliminação de Gauss:

- Eliminar a_{21}:

$$l_{21} = \frac{a_{21}}{u_{11}}$$

$$u_{22} = a_{22} - l_{21}u_{12}$$

$$u_{23} = a_{23} - l_{21}u_{13}$$

- Eliminar a_{31}:

$$l_{31} = \frac{a_{31}}{u_{11}}$$

$$u_{32} = a_{32} - l_{31}u_{12}$$

$$u_{33} = a_{33} - l_{31}u_{13} - l_{32}u_{23}$$

3. Resultado: $A = LU$

Exercícios

1. **Implemente** a função `lu_decomposition` em Python utilizando operações elementares.

```
def lu_decomposition(A):
    n = len(A)
    L = [[0.0] * n for _ in range(n)]
    U = [[0.0] * n for _ in range(n)]

    for i in range(n):
        L[i][i] = 1.0
        for j in range(i, n):
            U[i][j] = A[i][j] - sum(L[i][k] * U[k][j] for k in range(i))
        for j in range(i+1, n):
            L[j][i] = (A[j][i] - sum(L[j][k] * U[k][i] for k in range(i))) / U[i][i]
    return L, U
```

PERGUNTAS?