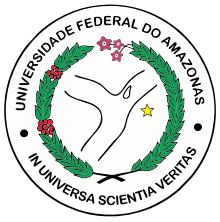


**UNIVERSIDADE FEDERAL DO AMAZONAS**

**PROGRAMA DE PÓS GRADUAÇÃO EM ENGENHARIA ELÉTRICA**

# IMPLEMENTAÇÃO DO ALGORITMO DO ROW ECHELON FORM

**Andevaldo da Encarnação Vitório**

**MANAUS-AM**

**2024**

Andevaldo da Encarnação Vitório

# IMPLEMENTAÇÃO DO ALGORITMO DO ROW ECHELON FORM

Este trabalho foi preparado como parte dos requisitos da disciplina *Sistemas Lineares* oferecida pelo Programa de Pós-graduação em Engenharia Elétrica da Universidade Federal do Amazonas.

Prof. Dr. João Edgar Chaves Filho

MANAUS-AM

2024

# Capítulo 1

# Algoritmo do Row Echelon Form

Este capítulo apresenta a descrição do código que realiza uma das operações essenciais de álgebra linear: a redução de matrizes à forma escalonada e à forma escalonada reduzida por linhas. Para isso, foi utilizada a linguagem *Python* e a sua biblioteca *NumPy*. A função principal é a *row_echelon_form*, que permite ao usuário escolher entre a forma escalonada e a forma escalonada reduzida. Além disso, uma função auxiliar chamada *show_matrix* é implementada para exibir as matrizes resultantes de forma formatada e de fácil visualização.

## 1.1 Descrição do Algoritmo

A redução de matrizes é um procedimento fundamental em álgebra linear, frequentemente utilizado para resolver sistemas de equações lineares, calcular o posto de uma matriz e determinar a invertibilidade de matrizes. O algoritmo recebe como entrada uma matriz $A$ e realiza operações elementares sobre as linhas da matriz para transformá-la em uma forma escalonada de linha. Opcionalmente, pode-se obter a forma escalonada reduzida de linha se o parâmetro *reduced* for verdadeiro. A seguir, detalharemos o funcionamento e a base matemática do algoritmo.

### 1.1.1 Definições

Uma matriz $A$ está na forma escalonada de linha (REF, do inglês *Row Echelon Form*) se satisfizer as seguintes condições:

- A primeira entrada não nula de cada linha (chamada de *pivô*) é 1;
- Os pivôs de cada linha aparecem à direita dos pivôs da linha anterior;

- Todas as linhas que contêm apenas zeros estão na parte inferior da matriz;

Além disso, uma matriz $A$ está na forma escalonada reduzida de linha (RREF, do inglês *Reduced* REF) se, além das condições para REF, satisfizer a seguinte condição: cada pivô é o único elemento não nulo em sua coluna.

## 1.2 Implementação em Python

Baseado nas definições do algoritmo, foi realizada a sua implementação em *Python*. O código implementado é apresentado a seguir.

```python
import numpy as np

def row_echelon_form(A, reduced=False):
"""
Reduces the matrix A to row echelon form or reduced row echelon form.

Parameters:
A: input matrix.
reduced: boolean, if True, returns the reduced row echelon form (RREF).
          If False, returns only the row echelon form (REF).

Returns:
A: matrix reduced to the desired form.
"""
A = np.array(
    A, dtype=float)  # Convert the matrix to float to ensure precision
rows, cols = A.shape

# Reduce the matrix to row echelon form (REF)
for col in range(min(rows, cols)):
  # Choose the pivot
  max_row = np.argmax(np.abs(A[col:rows, col])) + col
  if A[max_row, col] == 0:
    continue  # Skip the column if the pivot is zero

  # Swap the current row with the row containing the pivot
  A[[col, max_row]] = A[[max_row, col]]

  # Normalize the pivot row so that the pivot is 1
  A[col] = A[col] / A[col, col]

  # Eliminate the values below the pivot
  for i in range(col + 1, rows):
    A[i] -= A[i, col] * A[col]
```

```
35
36  # If reduced is True, reduce to the reduced row echelon form (RREF)
37  if reduced:
38    for col in range(min(rows, cols) - 1, -1, -1):
39      pivot_row = None
40      for row in range(rows):
41        if A[row, col] == 1:
42          pivot_row = row
43          break
44
45      if pivot_row is not None:
46        for row in range(pivot_row):
47          A[row] -= A[row, col] * A[pivot_row]
48
49  return A
```

Código 1.1: Implementação do algoritmo para obtenção da REF e RREF de uma matriz.

Inicialmente, a matriz de entrada $A$ é convertida para o tipo *float* para garantir precisão nas operações. Isso é necessário para evitar erros de arredondamento que podem ocorrer com operações inteiras. Em seguida, para cada coluna da matriz, o algoritmo seleciona o pivô como o maior valor absoluto na parte não processada da coluna. Esse processo é feito para melhorar a estabilidade numérica do algoritmo, garantindo que divisões por valores pequenos sejam minimizadas. Formalmente, o pivô $a_{ij}$ é escolhido de tal forma que:

$$|a_{ij}| = \max\{|a_{kj}|, k \geq i\} \tag{1.1}$$

Se o valor do pivô for zero, a coluna é ignorada, pois não pode ser usada para escalonamento.

Após selecionar o pivô, a linha que o contém é trocada com a linha atual. Essa troca é realizada para garantir que o pivô seja utilizado corretamente na eliminação das entradas abaixo dele. A linha contendo o pivô é normalizada, dividindo todos os seus elementos pelo valor do pivô, de forma que o pivô se torne 1:

$$A[i,:] = \frac{A[i,:]}{A[i,i]} \tag{1.2}$$

Esse passo é necessário para garantir que o elemento pivô seja igual a 1, como exigido na definição de REF. Para cada linha abaixo da linha do pivô, subtrai-se uma fração da linha do pivô de modo a zerar os elementos abaixo do pivô:

$$A[j,:] = A[j,:] - A[j,i] \cdot A[i,:] \quad \text{para} \quad j > i \tag{1.3}$$

Desta forma, é garantido que todos os elementos abaixo do pivô na mesma coluna sejam nulos.

Se o parâmetro *reduced* for verdadeiro, o algoritmo realiza um passo adicional: para cada pivô, ele elimina os valores acima do pivô de forma que o pivô seja o único elemento não nulo em sua coluna. Isso é feito subtraindo múltiplos da linha do pivô das linhas acima dela:

$$A[k,:] = A[k,:] - A[k,i] \cdot A[i,:] \quad \text{para} \quad k < i \tag{1.4}$$

Esse processo transforma a matriz na forma escalonada reduzida de linha.

### 1.2.1 Complexidade Computacional

A complexidade computacional deste algoritmo é aproximadamente $O(n^3)$, onde $n$ é o número de linhas ou colunas da matriz, o que torna o método adequado para matrizes de tamanho moderado. O fator cúbico provém da necessidade de realizar operações sobre todos os elementos abaixo (ou acima, no caso da forma reduzida) dos pivôs.

### 1.2.2 Exemplo de Uso

Consideremos a seguinte matriz $A$:

$$A = \begin{bmatrix} 1 & 3 & 3 & 8 & 5 \\ 0 & 1 & 3 & 10 & 8 \\ 0 & 0 & 0 & -1 & -4 \\ 0 & 0 & 0 & 2 & 8 \end{bmatrix} \tag{1.5}$$

Aplicando o algoritmo *row_echelon_form* a esta matriz, pode-se obter tanto a forma escalonada de linha quanto a forma escalonada reduzida de linha. Para isso, temos:

```
A = [[1, 3, 3, 8, 5],
     [0, 1, 3, 10, 8],
     [0, 0, 0, -1, -4],
     [0, 0, 0, 2, 8]]

# Para obter a forma escalonada de linha (REF)
echelon_form = row_echelon_form(A, reduced=False)
print("Forma Escalonada de Linha:")
print(echelon_form)

# Para obter a forma escalonada reduzida de linha (RREF)
reduced_echelon_form = row_echelon_form(A, reduced=True)
print("\nForma Escalonada Reduzida de Linha:")
```

```
print(reduced_echelon_form)
\end{verbatim}
```

Código 1.2: Exemplo de uso do algoritmo.

A execução desse código resultará na transformação da matriz $A$ em suas respectivas formas escalonadas. Quando o parâmetro `reduced=False`, o algoritmo retorna a matriz na forma escalonada de linha. Quando `reduced=True`, a matriz resultante será a forma escalonada reduzida de linha. Este exemplo ilustra a flexibilidade do algoritmo em lidar com diferentes formas de redução de matrizes.

# Capítulo 2

# Conclusão

O algoritmo apresentado é uma implementação simples, porém robusta, para a redução de matrizes à forma escalonada de linha e à forma escalonada reduzida de linha. Ele é amplamente utilizado em problemas de álgebra linear, como a solução de sistemas lineares, cálculo de determinantes e avaliação do posto de uma matriz. A precisão numérica é garantida pelo uso de operações em ponto flutuante e pela escolha inteligente dos pivôs. A versão opcional da redução à forma escalonada reduzida torna o algoritmo ainda mais versátil para aplicações que exigem uma forma mais simplificada da matriz.