Eliminação Gaussiana

Prof. Americo Cunha

Universidade do Estado do Rio de Janeiro – UERJ

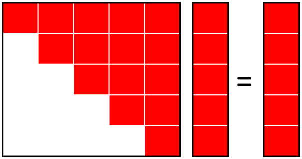
americo.cunha@uerj.br

www.americocunha.org

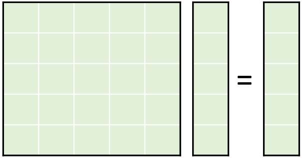
@AmericoCunhaJr @AmericoCunhaJr

@AmericoCunhaJr @AmericoCunhaJr

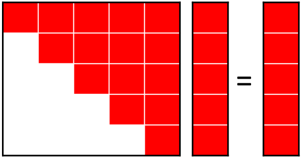
Num sistema linear triangular, existe um desacoplamento entre as equações, que facilita a solução!



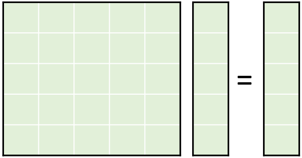
Num sistema linear “cheio”, não existe tal desacoplamento!



Num sistema linear triangular, existe um desacoplamento entre as equações, que facilita a solução!



Num sistema linear “cheio”, não existe tal desacoplamento!



Como proceder nesse caso?

Sistema $n \times n$ "cheio"

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

- A matriz $A \in \mathbb{R}^{n\times n}$ *é "cheia"*, i.e., *não tem uma estrutura especial com muitos zeros*;
- Em geral, nesse sistema linear as *equações são acopladas*, cada equação depende de todas as demais equações;
- O *método direto* padrão para resolver esse tipo de sistema linear é chamado *eliminação gaussiana*.

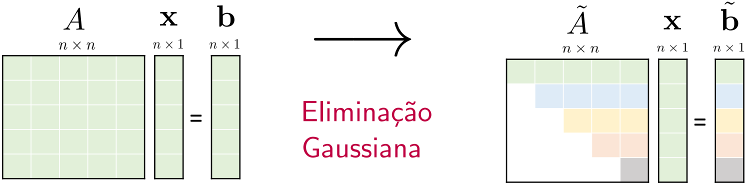
Eliminação Gaussiana

Ideia do método:

- Transformar o sistema “cheio” num sistema triangular, que seja equivalente ao sistema original (tenha a mesma solução);
- Resolver o sistema triangular por substituição.

Sistema Original

Sistema Equivalente
(mesma solução)

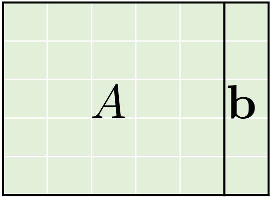


Como transformar um sistema “cheio” num triangular?

Através de uma sequência de *operações elementares*:

- trocar duas linhas de posição;
- multiplicar uma linha por uma constante não nula;
- somar um múltiplo de uma linha à outra linha.

O sistema original, em formato estendido, é alterado coluna por coluna até chegar num formato triangular superior equivalente:

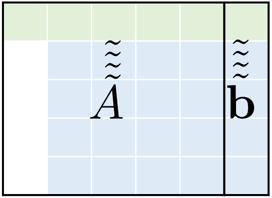


Como transformar um sistema "cheio" num triangular?

Através de uma sequência de *operações elementares*:

- trocar duas linhas de posição;
- multiplicar uma linha por uma constante não nula;
- somar um múltiplo de uma linha à outra linha.

O sistema original, em formato estendido, é alterado coluna por coluna até chegar num formato triangular superior equivalente:

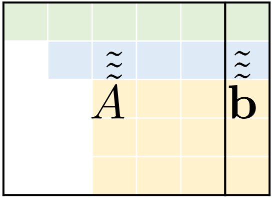


Como transformar um sistema “cheio” num triangular?

Através de uma sequência de *operações elementares*:

- trocar duas linhas de posição;
- multiplicar uma linha por uma constante não nula;
- somar um múltiplo de uma linha à outra linha.

O sistema original, em formato estendido, é alterado coluna por coluna até chegar num formato triangular superior equivalente:

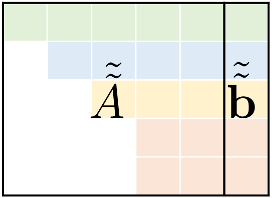


Como transformar um sistema “cheio” num triangular?

Através de uma sequência de *operações elementares*:

- trocar duas linhas de posição;
- multiplicar uma linha por uma constante não nula;
- somar um múltiplo de uma linha à outra linha.

O sistema original, em formato estendido, é alterado coluna por coluna até chegar num formato triangular superior equivalente:

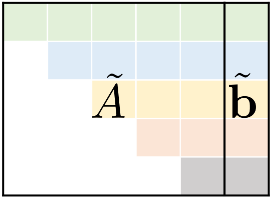


Como transformar um sistema “cheio” num triangular?

Através de uma sequência de *operações elementares*:

- trocar duas linhas de posição;
- multiplicar uma linha por uma constante não nula;
- somar um múltiplo de uma linha à outra linha.

O sistema original, em formato estendido, é alterado coluna por coluna até chegar num formato triangular superior equivalente:

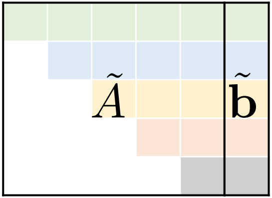


Como transformar um sistema “cheio” num triangular?

Através de uma sequência de *operações elementares*:

- trocar duas linhas de posição;
- multiplicar uma linha por uma constante não nula;
- somar um múltiplo de uma linha à outra linha.

O sistema original, em formato estendido, é alterado coluna por coluna até chegar num formato triangular superior equivalente:



Também é possível aplicar o algoritmo operando nas colunas.

O procedimento de eliminação

$$\left[\begin{array}{c|c} A^{(1)} & \mathbf{b}^{(1)} \end{array}\right] = \left[\begin{array}{ccccc|c} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & \cdots & a_{1n}^{(1)} & b_1^{(1)} \\ a_{21}^{(1)} & a_{22}^{(1)} & a_{23}^{(1)} & \cdots & a_{2n}^{(1)} & b_2^{(1)} \\ a_{31}^{(1)} & a_{32}^{(1)} & a_{33}^{(1)} & \cdots & a_{3n}^{(1)} & b_3^{(1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n1}^{(1)} & a_{n2}^{(1)} & a_{n3}^{(1)} & \cdots & a_{nn}^{(1)} & b_n^{(1)} \end{array}\right]$$

O procedimento de eliminação

Passo 1: anular todos os elementos abaixo do pivô $a_{11}^{(1)}$:

$$\left[\begin{array}{c|c} A^{(1)} & \mathbf{b}^{(1)} \end{array}\right] = \left[\begin{array}{ccccc|c} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & \cdots & a_{1n}^{(1)} & b_1^{(1)} \\ a_{21}^{(1)} & a_{22}^{(1)} & a_{23}^{(1)} & \cdots & a_{2n}^{(1)} & b_2^{(1)} \\ a_{31}^{(1)} & a_{32}^{(1)} & a_{33}^{(1)} & \cdots & a_{3n}^{(1)} & b_3^{(1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n1}^{(1)} & a_{n2}^{(1)} & a_{n3}^{(1)} & \cdots & a_{nn}^{(1)} & b_n^{(1)} \end{array}\right]$$

Para $i = 2, \ldots, n$, atualize as linhas segundo:

$$L_i^{(2)} \leftarrow L_i^{(1)} - \frac{a_{i\,1}^{(1)}}{a_{11}^{(1)}} L_1^{(1)}$$

O procedimento de eliminação

$$
\left[\begin{array}{c|c} A^{(2)} & \mathbf{b}^{(2)} \end{array}\right] = \left[\begin{array}{ccccc|c}
a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & \cdots & a_{1n}^{(1)} & b_1^{(1)} \\
0 & a_{22}^{(2)} & a_{23}^{(2)} & \cdots & a_{2n}^{(2)} & b_2^{(2)} \\
0 & a_{32}^{(2)} & a_{33}^{(2)} & \cdots & a_{3n}^{(2)} & b_3^{(2)} \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & a_{n2}^{(2)} & a_{n3}^{(2)} & \cdots & a_{nn}^{(2)} & b_n^{(2)}
\end{array}\right]
$$

O procedimento de eliminação

Passo 2: anular todos os elementos abaixo do pivô $a_{22}^{(2)}$:

$$\left[\begin{array}{c|c} A^{(2)} & \mathbf{b}^{(2)} \end{array}\right] = \left[\begin{array}{ccccc|c} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & \cdots & a_{1n}^{(1)} & b_1^{(1)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & \cdots & a_{2n}^{(2)} & b_2^{(2)} \\ 0 & a_{32}^{(2)} & a_{33}^{(2)} & \cdots & a_{3n}^{(2)} & b_3^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & a_{n2}^{(2)} & a_{n3}^{(2)} & \cdots & a_{nn}^{(2)} & b_n^{(2)} \end{array}\right]$$

Para $i = 3, ..., n$, atualize as linhas segundo:

$$L_i^{(3)} \leftarrow L_i^{(2)} - \frac{a_{i\,2}^{(2)}}{a_{22}^{(2)}} L_2^{(2)}$$

O procedimento de eliminação

$$\left[\begin{array}{c|c} A^{(3)} & \mathbf{b}^{(3)} \end{array}\right] = \left[\begin{array}{ccccc|c} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & \cdots & a_{1n}^{(1)} & b_1^{(1)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & \cdots & a_{2n}^{(2)} & b_2^{(2)} \\ 0 & 0 & a_{33}^{(3)} & \cdots & a_{3n}^{(3)} & b_3^{(3)} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & a_{n3}^{(3)} & \cdots & a_{nn}^{(3)} & b_n^{(3)} \end{array}\right]$$

O procedimento de eliminação

Passo 3: anular todos os elementos abaixo do pivô $a_{33}^{(3)}$:

$$\left[\begin{array}{c|c} A^{(3)} & \mathbf{b}^{(3)} \end{array}\right] = \left[\begin{array}{ccccc|c} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & \cdots & a_{1n}^{(1)} & b_1^{(1)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & \cdots & a_{2n}^{(2)} & b_2^{(2)} \\ 0 & 0 & a_{33}^{(3)} & \cdots & a_{3n}^{(3)} & b_3^{(3)} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & a_{n3}^{(3)} & \cdots & a_{nn}^{(3)} & b_n^{(3)} \end{array}\right]$$

Para $i = 4, \ldots, n$, atualize as linhas segundo:

$$L_i^{(4)} \leftarrow L_i^{(3)} - \frac{a_{i\,3}^{(3)}}{a_{33}^{(3)}} L_3^{(3)}$$

O procedimento de eliminação

... continua até a coluna $(n-1)$ ....

O procedimento de eliminação

$$\left[\begin{array}{c|c} A^{(n)} & \mathbf{b}^{(n)} \end{array}\right] = \left[\begin{array}{ccccc|c} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & \cdots & a_{1n}^{(1)} & b_1^{(1)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & \cdots & a_{2n}^{(2)} & b_2^{(2)} \\ 0 & 0 & a_{33}^{(3)} & \cdots & a_{3n}^{(3)} & b_3^{(3)} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & a_{nn}^{(n)} & b_n^{(n)} \end{array}\right]$$

O procedimento de eliminação

A *fórmula geral da eliminação gaussiana* é dada por

$$L_i^{(k+1)} \leftarrow L_i^{(k)} - \frac{a_{i\,k}^{(k)}}{a_{kk}^{(k)}} L_k^{(k)}$$

onde $k = 1, 2, \cdots, n-1$.

Algoritmo para a eliminação gaussiana

Input: A, $\mathbf{b}$

Compute the length of $\mathbf{b}$
for k=1:n-1 **do**
    for i=k+1:n **do**
        $l_{ik} = -a_{ik}/a_{kk}$
        for j=1:n **do**
            $a_{ij} = a_{ij} + l_{ik}\, a_{kj}$
        end for
        $b_i = b_i + l_{ik}\, b_k$
    end for
end for
Compute $\mathbf{x}$ with backward substitution
return

Output: $\mathbf{x}$

Esse algoritmo tem uma implementação pedagógica da eliminação gaussiana,

não é o mais eficiente do ponto de vista computacional.

Implementação em GNU Octave

```
function [x,A,b] = gauss(A,b)
   n = length(b);
   for k=1:n-1
      for i=k+1:n
         Lik = -A(i,k)/A(k,k);
         for j=1:n
            A(i,j) = A(i,j) + Lik*A(k,j);
         end
         b(i) = b(i) + Lik*b(k);
      end
   end
   x = backwardsub(A,b);
end
```

Experimento Computacional 1

$$\begin{bmatrix} 2 & 1 & -1 \\ -3 & -1 & 2 \\ -2 & 1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 8 \\ -11 \\ -3 \end{bmatrix} \qquad \longrightarrow \qquad \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \\ -1 \end{bmatrix}$$

```
>> A = [2 1 -1; -3 -1 2; -2 1 2]
>> b = [8; -11; -3]
>> [x,A,b] = gauss(A,b)
```

Experimento Computacional 2

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 4 & 3 \\ 3 & 2 & 1 & 4 \\ 4 & 3 & 2 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 10 \\ 10 \\ 10 \\ 10 \end{bmatrix} \qquad \longrightarrow \qquad \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

```
>> A = [1 2 3 4; 2 1 4 3; 3 2 1 4; 4 3 2 1]
>> b = [10; 10; 10; 10]
>> [x,A,b] = gauss(A,b)
```

Experimento Computacional 3

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 2 \\ 1 & 2 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \qquad \longrightarrow \qquad \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}$$

```
>>  A = [1 1 1; 1 1 2; 1 2 2]
>>  b = [1; 2; 1]
>>  [x,A,b] = gauss(A,b)
```

Qual o problema nesse último experimento?

Qual o problema nesse último experimento?

Eliminação Gaussiana:

Qual o problema nesse último experimento?

Eliminação Gaussiana:

$$\left[\begin{array}{ccc|c} 1 & 1 & 1 & 1 \\ 1 & 1 & 2 & 2 \\ 1 & 2 & 2 & 1 \end{array}\right]$$

Qual o problema nesse último experimento?

Eliminação Gaussiana:

$$\left[\begin{array}{ccc|c} 1 & 1 & 1 & 1 \\ 1 & 1 & 2 & 2 \\ 1 & 2 & 2 & 1 \end{array}\right] \qquad \begin{array}{l} L_2 \leftarrow L_2 - L_1 \\ L_3 \leftarrow L_3 - L_1 \end{array}$$

Qual o problema nesse último experimento?

Eliminação Gaussiana:

$$\left[\begin{array}{ccc|c} 1 & 1 & 1 & 1 \\ 1 & 1 & 2 & 2 \\ 1 & 2 & 2 & 1 \end{array}\right] \qquad \begin{array}{l} L_2 \leftarrow L_2 - L_1 \\ L_3 \leftarrow L_3 - L_1 \end{array}$$

$$\left[\begin{array}{ccc|c} 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{array}\right]$$

Qual o problema nesse último experimento?

Eliminação Gaussiana:

$$\left[\begin{array}{ccc|c} 1 & 1 & 1 & 1 \\ 1 & 1 & 2 & 2 \\ 1 & 2 & 2 & 1 \end{array}\right] \qquad \begin{array}{l} L_2 \leftarrow L_2 - L_1 \\ L_3 \leftarrow L_3 - L_1 \end{array}$$

$$\left[\begin{array}{ccc|c} 1 & 1 & 1 & 1 \\ 0 & \mathbf{0} & 1 & 1 \\ 0 & 1 & 1 & 0 \end{array}\right] \qquad L_3 \leftarrow L_3 - \frac{1}{\mathbf{0}} L_2$$

Divisão por zero, perigo!

Qual o problema nesse último experimento?

Eliminação Gaussiana:

$$\left[\begin{array}{ccc|c} 1 & 1 & 1 & 1 \\ 1 & 1 & 2 & 2 \\ 1 & 2 & 2 & 1 \end{array}\right] \qquad \begin{array}{l} L_2 \leftarrow L_2 - L_1 \\ L_3 \leftarrow L_3 - L_1 \end{array}$$

$$\left[\begin{array}{ccc|c} 1 & 1 & 1 & 1 \\ 0 & \mathbf{0} & 1 & 1 \\ 0 & 1 & 1 & 0 \end{array}\right] \qquad L_3 \leftarrow L_3 - \frac{1}{\mathbf{0}} L_2$$

Divisão por zero, perigo!

$$\left[\begin{array}{ccc|c} 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ \text{NaN} & \text{NaN} & -\infty & -\infty \end{array}\right]$$

Qual o problema nesse último experimento?

Eliminação Gaussiana:

$$\left[\begin{array}{ccc|c} 1 & 1 & 1 & 1 \\ 1 & 1 & 2 & 2 \\ 1 & 2 & 2 & 1 \end{array}\right] \qquad \begin{array}{l} L_2 \leftarrow L_2 - L_1 \\ L_3 \leftarrow L_3 - L_1 \end{array}$$

$$\left[\begin{array}{ccc|c} 1 & 1 & 1 & 1 \\ 0 & \mathbf{0} & 1 & 1 \\ 0 & 1 & 1 & 0 \end{array}\right] \qquad L_3 \leftarrow L_3 - \frac{1}{\mathbf{0}} L_2$$

Divisão por zero, perigo!

$$\left[\begin{array}{ccc|c} 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ \mathrm{NaN} & \mathrm{NaN} & -\infty & -\infty \end{array}\right]$$

Os demais cálculos são comprometidos por NaN e $-\infty$.

Para pensar em casa ...

Exercício computacional:

Pense num algoritmo mais eficiente (em termos de processamento e uso de memória) para implementar a eliminação gaussiana. Implemente esse algoritmo no ambiente GNU Octave.

Exercício computacional:

Pense num algoritmo para a eliminação gaussiana, que resolva o problema de ter um denominador nulo durante o processo de escalonamento da matriz estendida. Implemente esse algoritmo no ambiente GNU Octave.

Mais a frente nesse curso veremos como remediar essa questão do denominador nulo.

Qual o custo computacional de um sistema cheio?

Elim. Gauss. = Triangularização + Substituição

Qual o custo computacional de um sistema cheio?

Elim. Gauss. = Triangularização + Substituição

$$\texttt{flops}\,(\textbf{Triangularização}) \approx \underbrace{(n-1)+(n-2)+\cdots+1}_{\text{divisões}} + \underbrace{(n-1)^2+(n-2)^2+\cdots+1}_{\text{multiplicações}} + \underbrace{(n-1)^2+(n-2)^2+\cdots+1}_{\text{subtrações}}$$

Qual o custo computacional de um sistema cheio?

Elim. Gauss. = Triangularização + Substituição

$$\texttt{flops}\,(\textbf{Triangularização}) \approx \underbrace{(n-1)+(n-2)+\cdots+1}_{\text{divisões}} + \underbrace{(n-1)^2+(n-2)^2+\cdots+1}_{\text{multiplicações}} + \underbrace{(n-1)^2+(n-2)^2+\cdots+1}_{\text{subtrações}} \sim \frac{2}{3}\,n^3$$

Qual o custo computacional de um sistema cheio?

Elim. Gauss. = Triangularização + Substituição

$$\texttt{flops}\,(\textbf{Triangularização}) \approx \underbrace{(n-1)+(n-2)+\cdots+1}_{\text{divisões}} + \underbrace{(n-1)^2+(n-2)^2+\cdots+1}_{\text{multiplicações}} + \underbrace{(n-1)^2+(n-2)^2+\cdots+1}_{\text{subtrações}} \sim \frac{2}{3}\,n^3$$

Processamento:

Memória:

Qual o custo computacional de um sistema cheio?

$$\text{Elim. Gauss.} = \text{Triangularização} + \text{Substituição}$$

$$\begin{aligned}\texttt{flops}\,(\textbf{Triangularização}) \;\approx\;& \underbrace{(n-1)+(n-2)+\cdots+1}_{\text{divisões}} \;+\\ & \underbrace{(n-1)^2+(n-2)^2+\cdots+1}_{\text{multiplicações}} \;+\\ & \underbrace{(n-1)^2+(n-2)^2+\cdots+1}_{\text{subtrações}}\\ \sim\;& \frac{2}{3}\,n^3\end{aligned}$$

Processamento:

$$\texttt{flops}\,(\textbf{Gauss}) \sim \frac{2}{3}\,n^3 + n^2$$

Memória:

$$\texttt{mem}\,(\textbf{cheio}) = n^2 + 2\,n$$

Tempo de processamento para um sistema "cheio"

Intel Core i7 em 2011: 12×10^9 flops/sec
Intel Core i7 em 2021: 52×10^9 flops/sec

n	flops (Elim. Gauss.)	Tempo de CPU 2011	Tempo de CPU 2021
10	767	64 ns	14 ns
10^2	$6,7 \times 10^5$	56 μs	13 μs
10^3	$6,7 \times 10^8$	56 ms	13 ms
10^4	$6,7 \times 10^{11}$	56 s	13 s
10^5	$6,7 \times 10^{14}$	15 horas	4 horas
10^6	$6,7 \times 10^{17}$	643 dias	148 dias
10^7	$6,7 \times 10^{20}$	1,7k anos	406 anos
10^8	$6,7 \times 10^{23}$	1,7M anos	406k anos

https://en.wikipedia.org/wiki/FLOPS https://www.cpubenchmark.net

Uso de memória para um sistema “cheio”

1 double = 8 bytes

n	entradas	memória
10	120	1 kB
10^2	10×10^3	80 kB
10^3	1×10^6	8 MB
10^4	100×10^6	763 MB
10^5	10×10^9	75 GB
10^6	1×10^{12}	7 TB
10^7	100×10^{12}	727 TB
10^8	1×10^{16}	71 PB

Moral sobre sistemas lineares "cheios"

1. Sistemas lineares "cheios" são muito desafiadores, tanto em tempo de CPU quanto ao uso de memória;
2. Sistemas "cheios" que são intratáveis via métodos diretos podem ser atacados por métodos iterativos junto com estratégias de armazenamento otimizadas;
3. Na maioria das aplicações práticas as matrizes tem estruturas especiais (e.g. simétrica, em banda etc), que podem ser exploradas para reduzir tempo de CPU e uso de memória;
4. O método numérico a ser utilizado na solução do sistema linear, bem como a estratégia de armazenamento, devem ser escolhidos com sabedoria!

Como citar esse material?

A. Cunha, *Eliminação Gaussiana*,
Universidade do Estado do Rio de Janeiro – UERJ, 2021.

@AmericoCunhaJr @AmericoCunhaJr

@AmericoCunhaJr @AmericoCunhaJr