

# Alguns Sistemas Lineares Especiais

Prof. Americo Cunha

Universidade do Estado do Rio de Janeiro – UERJ

[americo.cunha@uerj.br](mailto:americo.cunha@uerj.br)

[www.americocunha.org](http://www.americocunha.org)



@AmericoCunhaJr



@AmericoCunhaJr



@AmericoCunhaJr



@AmericoCunhaJr



Sistemas lineares de grande porte são  **muito caros!**

$$A \mathbf{x} = \mathbf{b}$$

\$ \$ \$ \$ \$

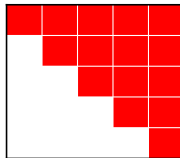
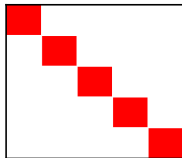
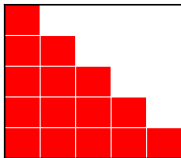


Sistemas lineares de grande porte são  **muito caros!**

$$A x = b$$

\$ \$ \$ \$ \$

Mas em alguns sistemas a matriz tem uma **estrutura especial**, que facilita o cálculo de uma solução!



## Sistema $n \times n$ em formato diagonal

$$\begin{bmatrix} a_{1,1} & & & \\ & a_{2,2} & & \\ & & \ddots & \\ & & & a_{n,n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

- *Sistema não singular*, i.e.,  $\det A \neq 0$ ;
- A matriz  $A \in \mathbb{R}^{n \times n}$  é *diagonal*, i.e., todas as entradas fora da diagonal principal são iguais a zero ( $a_{i,j} = 0$ ,  $i \neq j$ ).



## Sistema $n \times n$ em formato diagonal

$$\left\{ \begin{array}{rcl} a_{1,1} x_1 & = & b_1 \\ & a_{2,2} x_2 & = b_2 \\ & & \ddots \\ & & a_{n,n} x_n = b_n \end{array} \right.$$

- A representação acima corresponde a um *sistema desacoplado*, pois cada equação é independente das demais (as equações tem apenas uma variável);
- A ordem em que as equações são resolvidas não é relevante!

$$x_i = \frac{b_i}{a_{i,i}}, \quad i = 1, \dots, n$$



## Sistema $n \times n$ em formato triangular superior

$$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \cdots & a_{1,n} \\ & a_{2,2} & a_{2,3} & \cdots & a_{2,n} \\ & & \ddots & \ddots & \vdots \\ & & & a_{n-1,n-1} & a_{n-1,n} \\ & & & & a_{n,n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_{n-1} \\ b_n \end{bmatrix}$$

- *Sistema não singular*, i.e.,  $\det A \neq 0$ ;
- A matriz  $A \in \mathbb{R}^{n \times n}$  é *triangular superior*, i.e., todas as entradas abaixo da diagonal principal são iguais a zero ( $a_{i,j} = 0$ ,  $i > j$ ).



## Sistema $n \times n$ em formato triangular superior

$$\left\{ \begin{array}{ccccccc} a_{1,1} x_1 + a_{1,2} x_2 + \cdots + a_{1,n-1} x_{n-1} + a_{1,n} x_n & = & b_1 \\ & a_{2,2} x_2 + \cdots + a_{2,n-1} x_{n-1} + a_{2,n} x_n & = & b_2 \\ & & \ddots & & \vdots & & \vdots \\ & & & a_{n-1,n-1} x_{n-1} + a_{n-1,n} x_n & = & b_{n-1} \\ & & & & a_{n,n} x_n & = & b_n \end{array} \right.$$

- A representação acima corresponde a um *sistema desacoplado de baixo para cima*, pois cada equação depende apenas das equações posteriores;
- As equações devem ser resolvidas de baixo para cima, através de um processo de *substituição regressiva*:

$$x_i = \frac{b_i - \sum_{j=i+1}^n a_{i,j} x_j}{a_{i,i}}, \quad i = n, \dots, 1$$



## Sistema $n \times n$ em formato triangular inferior

$$\begin{bmatrix} a_{1,1} & & & & \\ a_{2,1} & a_{2,2} & & & \\ \vdots & \vdots & \ddots & & \\ a_{n-1,1} & a_{n-1,2} & \cdots & a_{n-1,n} & \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n-1} & a_{n,n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_{n-1} \\ b_n \end{bmatrix}$$

- *Sistema não singular*, i.e.,  $\det A \neq 0$ ;
- A matriz  $A \in \mathbb{R}^{n \times n}$  é *triangular inferior*, i.e., todas as entradas acima da diagonal principal são iguais a zero ( $a_{i,j} = 0$ ,  $i < j$ ).





## Sistema $n \times n$ em formato triangular inferior

$$\left\{ \begin{array}{lcl} a_{1,1} x_1 & & = b_1 \\ a_{2,1} x_1 + a_{2,2} x_2 & & = b_2 \\ \vdots & \vdots & \vdots \\ a_{n-1,1} x_1 + a_{n-1,2} x_2 + \cdots + a_{n-1,n} x_{n-1} & & = b_{n-1} \\ a_{n,1} x_1 + a_{n,2} x_2 + \cdots + a_{n,n-1} x_{n-1} + a_{n,n} x_n & & = b_n \end{array} \right.$$

- A representação acima corresponde a um *sistema desacoplado de cima para baixo*, pois cada equação depende apenas das equações anteriores;
- As equações devem ser resolvidas de cima para baixo, através de um processo de *substituição progressiva*:

$$x_i = \frac{b_i - \sum_{j=1}^{i-1} a_{i,j} x_j}{a_{i,i}}, \quad i = 1, \dots, n$$



# Algoritmo para um sistema triangular superior

Input:  $A$ ,  $b$

```
1: Compute the length of  $b$ 
2: Allocate memory for  $x$ 
3: for  $i=n:1$  do
4:    $x_i = b_i$ 
5:   for  $j=i+1:n$  do
6:      $x_i = x_i + a_{ij} x_j$ 
7:   end for
8:    $x_i = x_i / a_{ii}$ 
9: end for
10: return
```

Output:  $x$

Esse algoritmo tem uma implementação pedagógica da substituição regressiva,  
não é o mais eficiente do ponto de vista computacional.

Exercício computacional:

Adapte esse algoritmo para um sistema triangular inferior.



# Implementação em GNU Octave

```
function x = backsubsub(A,b)
    n = length(b);
    x = zeros(n,1);
    for i = n:-1:1
        x(i) = b(i);
        for j = i+1:n
            x(i) = x(i) - A(i,j)*x(j);
        end
        x(i) = x(i)/A(i,i);
    end
end
```



# Implementação em GNU Octave

```
function x = forwardsub(A,b)
    n = length(b);
    x = zeros(n,1);
    for i = 1:n
        x(i) = b(i);
        for j = 1:i-1
            x(i) = x(i) - A(i,j)*x(j);
        end
        x(i) = x(i)/A(i,i);
    end
end
```



# Experimento Computacional 1

$$\begin{bmatrix} 1 & 1 & \cdots & 1 & 1 \\ & 1 & \cdots & 1 & 1 \\ & & \ddots & & \vdots \\ & & & 1 & 1 \\ & & & & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} n \\ n-1 \\ \vdots \\ 2 \\ 1 \end{bmatrix} \Rightarrow \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ 1 \end{bmatrix}$$



```
>> n = 5; A = triu(ones(n,n)); b = A*ones(n,1);  
>> tic; x = backsubsub(A,b); toc
```

$$\begin{bmatrix} 1 & & & & \\ 1 & 1 & & & \\ \vdots & & \ddots & & \\ 1 & 1 & \cdots & 1 & \\ 1 & 1 & \cdots & 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ \vdots \\ n-1 \\ n \end{bmatrix} \Rightarrow \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ 1 \end{bmatrix}$$



```
>> n = 5; A = tril(ones(n,n)); b = A*ones(n,1);  
>> tic; x = forwardsub(A,b); toc
```



# Qual o custo computacional de um sistema triangular?



# Qual o custo computacional de um sistema triangular?

$$\text{flops}(\mathbf{triangular}) = 1 + 3 + 5 + \cdots + 2n - 1$$



# Qual o custo computacional de um sistema triangular?

$$\text{flops}(\mathbf{triangular}) = \underbrace{1 + 3 + 5 + \cdots + 2n - 1}_{\text{Soma de uma P.A.}}$$





# Qual o custo computacional de um sistema triangular?

$$\begin{aligned}\text{flops}(\mathbf{triangular}) &= \underbrace{1 + 3 + 5 + \cdots + 2n - 1}_{\text{Soma de uma P.A.}} \\ &= \frac{1}{2} n (2n - 1 + 1)\end{aligned}$$



# Qual o custo computacional de um sistema triangular?

$$\begin{aligned}\text{flops}(\mathbf{triangular}) &= \underbrace{1 + 3 + 5 + \dots + 2n - 1}_{\text{Soma de uma P.A.}} \\ &= \frac{1}{2} n (2n - 1 + 1) \\ &= n^2\end{aligned}$$



# Qual o custo computacional de um sistema triangular?

$$\begin{aligned}\text{flops}(\mathbf{triangular}) &= \underbrace{1 + 3 + 5 + \cdots + 2n - 1}_{\text{Soma de uma P.A.}} \\ &= \frac{1}{2} n (2n - 1 + 1) \\ &= n^2\end{aligned}$$

Processamento:

Memória:



# Qual o custo computacional de um sistema triangular?

$$\begin{aligned}\text{flops}(\mathbf{triangular}) &= \underbrace{1 + 3 + 5 + \cdots + 2n - 1}_{\text{Soma de uma P.A.}} \\ &= \frac{1}{2} n (2n - 1 + 1) \\ &= n^2\end{aligned}$$

Processamento:

$$\text{flops}(\mathbf{triangular}) \sim n^2$$

Memória:

$$\text{mem}(\mathbf{triangular}) = n(n-1)/2 + 2n$$



# Qual o custo computacional de um sistema triangular?

$$\begin{aligned}\text{flops}(\mathbf{triangular}) &= \underbrace{1 + 3 + 5 + \cdots + 2n - 1}_{\text{Soma de uma P.A.}} \\ &= \frac{1}{2} n (2n - 1 + 1) \\ &= n^2\end{aligned}$$

Processamento:

$$\text{flops}(\mathbf{triangular}) \sim n^2$$

$$\text{flops}(\mathbf{diagonal}) \sim n$$

Memória:

$$\text{mem}(\mathbf{triangular}) = n(n - 1)/2 + 2n$$

$$\text{mem}(\mathbf{diagonal}) = 3n$$



# Tempo de processamento para um sistema triangular

Intel Core i7 em 2011:  $12 \times 10^9$  FLOPS

Intel Core i7 em 2021:  $52 \times 10^9$  FLOPS

FLOPS = flops por segundo

n	flops	Tempo de CPU	
		2011	2021
10	$10^2$	8 ns	2 ns
$10^2$	$10^4$	8 $\mu$ s	2 $\mu$ s
$10^3$	$10^6$	83 $\mu$ s	19 $\mu$ s
$10^4$	$10^8$	8 ms	2 ms
$10^5$	$10^{10}$	0.8 s	0.2 s
$10^6$	$10^{12}$	83 s	19 s
$10^7$	$10^{14}$	139 min	53 min
$10^8$	$10^{16}$	9,65 dias	2,23 dias

<https://en.wikipedia.org/wiki/FLOPS>

<https://www.cpubenchmark.net>

# Uso de memória para um sistema triangular

1 double = 8 bytes

n	entradas	memória
10	65	0,5 kB
$10^2$	$5 \times 10^3$	40 kB
$10^3$	$500 \times 10^3$	490 kB
$10^4$	$50 \times 10^6$	48 MB
$10^5$	$5 \times 10^9$	5 GB
$10^6$	$500 \times 10^9$	465 GB
$10^7$	$50 \times 10^{12}$	45 TB
$10^8$	$5 \times 10^{15}$	5PB



# Moral sobre sistemas lineares especiais


1. **Sistemas lineares de grande porte com uma estrutura especial (triangular ou diagonal) podem ser resolvidos num tempo de processamento aceitável;**
2. **Sistemas lineares de grande porte demandam uma enorme quantidade de memória, mesmo quando tem estrutura triangular. Na prática isso pode ser um fator limitante mais significativo que o tempo de processamento;**
3. **O método numérico a ser utilizado na solução do sistema linear, bem como a estratégia de armazenamento, devem ser escolhidos com sabedoria!**






## Para pensar em casa ...

### Exercício computacional:

Pense num algoritmo eficiente (em termos de processamento e uso de memória) para resolver um sistema diagonal. Implemente esse algoritmo no ambiente GNU Octave. 

### Exercício computacional:

Você consegue pensar num algoritmo otimizado para fazer a substituição progressiva/regressiva? Implemente esses dois algoritmos otimizados no ambiente GNU Octave. 

Dica:

Use o produto matricial para computar a soma.



## Como citar esse material?

A. Cunha, *Alguns Sistemas Lineares Especiais*,  
Universidade do Estado do Rio de Janeiro – UERJ, 2021.



 @AmericoCunhaJr

 @AmericoCunhaJr

 @AmericoCunhaJr

 @AmericoCunhaJr

Essas notas de aula podem ser compartilhadas nos termos da licença Creative Commons BY-NC-ND 3.0, com propósitos exclusivamente educacionais.

