

# I. Erros, condicionamento e estabilidade numérica

---

- O conjunto  $\mathbb{R}$  dos números reais é **infinito**, **contínuo** e **ilimitado**. O subconjunto  $\mathcal{F}$  dos números que se podem representar exatamente num computador digital é **finito**, **discreto** e **limitado**.
- Seja  $[x_-, x_+]$  um intervalo real onde  $x_+$  é o **sucessor** de  $x_-$  em  $\mathcal{F}$ ; um número  $x$  de  $]x_-, x_+[$  é representado por  $x_-$  ou  $x_+$  [nota: usaremos  $fl(x)$  para representar o valor arredondado de  $x$ ].
- $|x - fl(x)|$  erro **absoluto** devido ao arredondamento
- $\frac{|x - fl(x)|}{|x|}$  erro **relativo** devido ao arredondamento [ $x$  não nulo]
- Os erros de arredondamento podem ter efeitos catastróficos.

# Erros catastróficos: exemplo

- No dia 25 de Fevereiro de 1991, durante o conflito que resultou da invasão do Kuwait por tropas iraquianas (Guerra do Golfo), uma bateria de mísseis norte-americanos Patriot cuja missão era a de proteger uma instalação militar em Daharan, na Arábia Saudita, falhou a interceptação de um míssil Scud, lançado pelas forças militares iraquianas. O Scud atingiu um aquartelamento das tropas americanas provocando 28 mortos e um número elevado de feridos.
- A origem do problema foi a propagação de um erro de arredondamento no computador da bateria dos mísseis Patriot.
- O sistema de controle do Patriot usava como unidade de tempo a décima parte do segundo; o número 0.1 não tem representação binária exata. O Patriot tinha registos de ponto fixo com 24 bits:

$$fl(0.1) = 0.0001100110011001100110011001100 \dots$$

Erro de arredondamento (absoluto)

$$|0.1 - fl(0.1)| = 2^{-24} + 2^{-25} + 2^{-28} \dots \approx 9.5 * 10^{-8}$$

# Continuação do exemplo

---

- É este pequeno erro  $9.5 * 10^{-8}$  que se vai propagar e causar o problema. A bateria estava ligada há cerca de 100 horas.
- Unidades (em décimos de segundo) de tempo: 3 600 000 (=100 \*60 \*60 \*10)
- $3\,600\,000 * 9.5 * 10^{-8}$  segundos  $\approx 3.4$  segundos
- O Scud tinha uma velocidade de 1.676 metros por segundo e portanto percorria mais de 500 metros num intervalo de tempo de 0.34 segundos. Este facto fez com que o Patriot falhasse a interceção do míssil iraquiano.

Este desastre e outros também causados por problemas de software numérico podem ser encontrados em <http://www.math.psu.edu/dna/disasters>

# Mais exemplos de erros de arredondamento

---

- Em aritmética exacta, as condições  $x > 0$  e  $1 + x > 1$  são equivalentes. Verifica que no Matlab

```
>> 2^-53 > 0
```

dá o valor lógico 1 (a proposição é verdadeira) mas

```
>> 1 + 2^-53 > 1
```

dá o valor lógico 0 (a proposição é falsa).

# Mais exemplos de erros de arredondamento

---

- O Matlab não consegue calcular o valor exato da soma de 10 parcelas todas iguais a 0.1

```
>> x = 0.1 * ones(10, 1)
```

[vector com 10 entradas todas iguais 0.1; em geral, >> ones(m; n) produz uma matriz com m linhas e n colunas e entradas todas iguais à unidade].

```
>> sum(x)           % soma todas as entrdras do vetor x
```

produz o resultado **1.0000** com um erro que é dado por

```
>> ans-1
```

aproximadamente igual a **-1.1102e-16**

# Mais exemplos de erros de arredondamento

---

- A adição e multiplicação de números reais são associativas e a multiplicação é distributiva em relação à adição mas, no Matlab, com

```
>> x = 0.1; y = 0.3; z = 0.7;
```

as condições

$$(x + z) + y == x + (z + y)$$
$$(x * y) * z == x * (y * z)$$
$$x * (y + z) == (x * y) + (x * z)$$

produzem no Matlab o valor lógico 0 (falso). Verifica.

# Cálculo numérico *versus* cálculo simbólico

---

- *Um sistema de cálculo simbólico representa os números racionais na forma de um quociente de dois inteiros e opera com eles usando as regras aritméticas apropriadas. O preço que se paga por isto é que o sistema usa mais memória para a representação dos números e a aritmética é "mais pesada".*
- Num sistema de cálculo simbólico obtém-se

$$\sum_{n=1}^{100} \frac{1}{n} = \frac{14\ 466\ 636\ 279\ 520\ 351\ 160\ 221\ 518\ 043\ 104\ 131\ 447\ 711}{2788\ 815\ 009\ 188\ 499\ 086\ 581\ 352\ 357\ 412\ 492\ 142\ 272}$$

e a soma dos inversos aritméticos dos 200 primeiros números inteiros positivos produz um numerador e denominador com 90 algarismos.

- Cálculo numérico e cálculo simbólico são ferramentas, ambas com vantagens e inconvenientes, que se complementam. O cálculo numérico continua a ser mais usado na computação científica mas existem códigos híbridos [numérico+simbólico].

# Cálculo numérico *versus* cálculo algébrico

---

No Matlab existe a Toolbox de cálculo simbólico.

*Exemplo 1*

```
>> sym(1/2)+sym(1/3)  
ans = 5/6
```

*Exemplo 2*

```
>> x=sym(1./(1:100))  
x = [1, 1/2, 1/3, 1/4, 1/5, 1/6,..., 1/100]
```

```
>> sum(x)  
ans=14466636279520351160221518043104131447711/2788815009188499  
086581352357412492142272
```

```
>> double(ans)  
ans = 5.187377517639621
```



# O ponto flutuante

---

- Na representação de números em **ponto flutuante**, um número  $x$  representa-se por

$$x = \pm m * \beta^e \quad (1)$$

onde  $m$  é a mantissa e  $e$  (inteiro positivo ou negativo) é o expoente.

- Valores típicos de  $\beta$  são 2 (sistema binário), 8 (sistema octal), 10 (decimal) e 16 (hexadecimal).
- Para o mesmo número  $x$  há muitas representações da forma (1). Por exemplo,

$$x = 0.2335 = 0.02335 * 10^1 = 23.35 * 10^{-2} = \dots$$

Obviamente, para adicionar dois números há que garantir que as representações têm o mesmo expoente e, neste caso, adicionam-se as respetivas mantissas.

- Exemplos:

$$(1.451 * 10^0) + (2.70145 * 10^{-3}) = (1.451 * 10^0) + (0.00270145 * 10^0) = 1.45370145 * 10^0$$

$$(1.00101 * 2^0) + (1.1001 * 2^{-4}) = (1.00101 * 2^0) + (0.00011001 * 2^0) = 1.01000001 * 2^0$$

# A norma IEEE754

---

- Os fabricantes de computadores têm adotado sistemas que diferem em muitos aspectos (base do sistema, número de dígitos na mantissa e no expoente, regras de arredondamento, etc.)
- Para os mesmos cálculos, diferentes computadores (ou máquinas de calcular) podem produzir resultados que não são exatamente iguais (podem até ser muito diferentes).
- No passado foi feito um esforço de uniformização que culminou com a publicação em 1985, da chamada "norma IEEE 754" para o sistema binário cujas especificidades apresentamos resumidamente.

formato simples (32 bits):

sinal (1 bit), expoente (8 bits), mantissa (23 bits)

formato duplo (64 bits):

sinal (1 bit), expoente (11 bits), mantissa (52 bits)

# A norma IEEE 754

---

- ▣ Representação normalizada: o primeiro bit da mantissa é igual a 1

$$\pm (1.b_{-1}b_{-2} \dots b_{-52}) * 2^e$$

- ▣ Exemplo

$$(0.1)_{10} = (0.0001100110011001100110011001100 \dots)_2$$

tem a representação normalizada (com 32 bits)

$$(\mathbf{1.10011001100110011001100}) * 2^{-4}$$

O bit à esquerda do ponto é o **bit implícito** (perfaz o total de 24 bits na mantissa).

# A norma IEEE754 (sobre o **epsilon**)

---

No formato duplo, o sucessor do número 1

$$1 = (1.0 \dots 00) * 2^0$$

(todos os bits iguais a 0, exceto o bit implícito) é o número que tem a representação

$$(1.0 \dots 01) * 2^0 = 1 + 2^{-52}$$

Esta distância do número 1 ao seu sucessor é de grande importância, como veremos mais adiante. É uma das constantes definidas no Matlab

```
>> eps
```

```
ans =
```

```
2.2204e-16
```

```
>> eps==2^-52
```

```
ans = logical 1
```

# Percorrendo $\mathcal{F}$

---

- Se o sucessor de 1 é  $1+\text{eps}$ , também o sucessor de  $1+\text{eps}$  é  $1+2*\text{eps}$ . Afinal, em geral, para obter o sucessor de um número de positivo de  $\mathcal{F}$  adiciona-se uma unidade na última posição da mantissa.
- Será que cada número dista do seu sucessor esta quantidade  $\text{eps}$ ?
- Isso só é verdade para os números com expoente  $e=0$ , isto é, números de  $\mathcal{F}$  entre 1 e 2. Por exemplo, o sucessor de

$$2 = (1.0 \dots 00) * 2^1$$

(nota: observe-se que todas as potências de 2 têm a mesma mantissa)

é 
$$(1.0 \dots 01) * 2^1 = (1 + \text{eps}) * 2 = 2 + 2\text{eps}$$

- A distância entre um número  $x_-$  e o seu sucessor  $x_+$  depende do expoente  $e$  de  $x_-$ . Se

$$x_- = \pm (1.b_{-1}b_{-2} \dots b_{-52}) * 2^e$$

é 
$$|x_+ - x_-| = 2^{e-52}$$

# Majoração do erro absoluto de arredondamento

---

□ Se  $x_- < x < x_+$   
resulta da expressão anterior que

$$|x - fl(x)| < 2^{e-52} \quad (2)$$

□  $2^{e-52}$  é o majorante do erro absoluto devido ao arredondamento

□ Para números grandes este erro absoluto pode ser grande;

Exemplo: se  $x$  tem expoente  $e=53$ , de (2) resulta

$$|x - fl(x)| < 2$$

o que mostra que o erro absoluto pode ser quase igual a 2 (e ainda maior do que isto para expoentes maiores).

[nota: este exemplo também mostra que há muitos números inteiros que não pertencem a  $\mathcal{F}$ ]

# Majoração do erro relativo de arredondamento

---

□ Com  $x = \pm (1.b_{-1}b_{-2} \dots b_{-52}) * 2^e$

resulta de (2) que  $\frac{|x - fl(x)|}{|x|} < \frac{2^{e-52}}{|x|} < 2^{-52}$  (por ser  $|x| > 2^e$ )

□ Tem-se

$$\frac{|x - fl(x)|}{|x|} < \text{eps} \quad (3)$$

isto é, o erro relativo é majorado por eps.

□ Ao contrário do erro absoluto, o erro relativo não depende da grandeza do número que é arredondado. Por outras palavras, o sistema de ponto flutuante representa números grandes e pequenos com a mesma precisão.

# A norma IEEE754 (expoentes)

---

- No formato simples, os oito bits reservados para o expoente permitem obter  
 $2^8 = 256$   
números positivos diferentes, desde 0 até 255.
- E para representar expoentes negativos (para números inferiores à unidade, em valor absoluto)?
- Não se usa um bit reservado para o sinal do expoente.
- *bias exponent* - expoente enviesado: para obter o verdadeiro expoente, o hardware subtrai 127.
- As representações 00000000 e 11111111 são usadas para situações especiais, número desnormalizado e overflow, resp.
- Formato simples  $-126 \leq e \text{ (inteiro)} \leq 127$
- Formato duplo  $-1022 \leq e \text{ (inteiro)} \leq 1023$



# O maior número representável na norma IEEE 754

$$\begin{aligned} & (\textbf{1.11})^* 2^{1023} \\ & = 2^{1023} + 2^{1022} + \dots + 2^{971} \end{aligned}$$

```
>> realmax
```

ans =

1.7977e+308

```
>> x = 0; for k = 971 : 1023; x = x + 2^k; end; x == realmax
```

ans =

1

# Overflow

---

Uma expressão para calcular realmax

$$\square \quad m = 1 + 2^{-1} + 2^{-2} + \dots + 2^{-51} + 2^{-52}$$

$$2m = 2 + 1 + 2^{-1} + \dots + 2^{-51}$$

$$m = 2 - 2^{-52}$$

$$\square \quad \text{realmax} = (2 - 2^{-52}) * 2^{1023} = 2^{1024} - 2^{971}$$

```
>> 2^1024-2^971==realmax  
ans = logical 0
```

```
>> (2-2^-52)*2^1023==realmax  
ans = logical 1
```

```
>> 2^1024  
ans = Inf
```

# O realmin

---

**O menor número normalizado é  $(1.0 \dots 0) \cdot 2^{-1022}$**

```
>> 2^-1022==realmin
```

```
ans = logical 1
```

```
>> realmin
```

```
ans =
```

```
2.2251e-308
```

# Underflow gradual

---

**O Matlab representa números cujo valor absoluto é menor do que realmin**

```
>> 2^-1023/2^-1024
```

```
ans =2
```

$$2^{-1023} = (0.10 \dots 00) * 2^{-1022}$$

$$2^{-1024} = (0.010 \dots 00) * 2^{-1022}$$

...

$$2^{-1074} = (0.000 \dots 01) * 2^{-1022}$$

```
>> 2^-1075
```

```
ans =0
```

Nota: números menores do que realmin são representados com menor precisão (são armazenados menos bits significativos)

# Arredondamento para o mais próximo

---

**Para o mais próximo ('default' no Matlab)**

```
x=1; x+eps/3 ==x
```

```
ans = logical 1
```

```
-----
```

```
>> x=1/2; x+eps/3==x
```

```
ans = logical 0
```

```
-----
```

```
>> x=1; x+eps/2==x
```

```
ans = logical 1
```

# Quatro modos de arredondamento

---

**Para a direita**

```
>> system_dependent('setround','Inf')
```

**Para a esquerda**

```
>> system_dependent('setround','-Inf')
```

**Na direção de zero (corte)**

```
>> system_dependent('setround','0')
```

**Para o mais próximo (´default´)**

```
>> system_dependent('setround','nearest')
```

# Soma com diferentes arredondamentos (1)

---

```
% esta script calcula o valor da soma de n números gerados
% aleatoriamente entre -1 e 1, usando os quatro modos de
% arredondamento previstos na norma IEEE

x=2*(rand(n,1)-0.5);
% rand(n,1) gera um vetor coluna com n entradas entre 0 e 1
system_dependent('setround',-Inf)
sEsq=sum(x)
% valor da soma calculada com arredondamento para -Inf
system_dependent('setround',0)
sZero=sum(x)
% valor da soma calculada com arredondamento para 0
system_dependent('setround',0.5)
sPro=sum(x)
% valor da soma calculada com arredondamento para o mais próximo
system_dependent('setround',Inf)
sDir=sum(x)
% valor da soma calculada com arredondamento para +Inf
```

# Soma com diferentes arredondamentos (2)

---

```
>> n=1000; format long; quatro_somas
```

O resultado exato está no intervalo [**sEsq**, **sDir**]

```
>> format long, n=100; quatro_somas
```

```
...
```

```
>> sDir-sEsq
```

```
ans =
```

```
2.131628207280301e-14
```

**O erro cresce com o número n de parcelas mas de forma moderada**



# Algarismos significativos

---

Nas representações seguintes o número  $\pi$  é aproximado com o mesmo número de algarismos significativos (neste caso, com 3 algarismos significativos)

$$3.14 * 10^0$$

$$314 * 10^{-2}$$

$$0.00314 * 10^3$$

Enfatiza-se que na última representação os zeros à direita do ponto decimal não são algarismos significativos.

- Na representação normalizada (norma IEEE 754)

$$\pm (1.b_{-1}b_{-2} \dots b_{-52}) * 2^e$$

os números têm todos 53 bits significativos independentemente da sua grandeza (expressa pelo expoente  $e$ )

# Algarismos significativos corretos

---

Se

$$\bar{x} = (0.d_{-1}d_{-2} \dots d_{-t}) * 10^e, \quad d_{-1} > 0$$

(mantissa com  $t$  algarismos) então  $|x - \bar{x}| < 10^{e-t}$ , qualquer que seja o modo de arredondamento. No **arredondamento para o mais próximo**

$$|x - \bar{x}| \leq \frac{10^{e-t}}{2}$$

e o majorante para o erro relativo é, por ser  $|x| \geq d_{-1} * 10^{e-1}$ ,

$$\frac{|x - \bar{x}|}{|x|} \leq \frac{1}{2} \frac{10^{e-t}}{10^{e-1}} = \frac{1}{2} 10^{-t+1} = 5 * 10^{-t}$$

**Dizemos neste caso que  $\bar{x}$  aproxima  $x$  com  $t$  algarismos significativos corretos**

# Exemplo no Matlab

---

$$\pi = 3.1415926535 \dots$$

Com **t** algarismos significativos corretos:

□  $t=3, \quad \bar{\pi} = 3.14$

>  $(\pi - 3.14)/\pi$  dá  $5.0696e-04$  ( $< 5 \cdot 10^{-3}$ )

□  $t=4, \quad \bar{\pi} = 3.142$  (e não **3.141**)

>>  $(\pi - 3.141)/\pi$  dá  $1.8865e-04$  ( $< 5 \cdot 10^{-4}$ )

□  $t=5, \quad \bar{\pi} = 3.1416$  (e não **3.1415**)

>>  $(\pi - 3.1416)/\pi$  dá  $-2.3384e-06$  ( $< 5 \cdot 10^{-5}$ )

# O cancelamento subtrativo (1)

---

Perda de algarismos significativos na subtração de números

## Exemplo

Sejam  $\bar{x} = 1.43275$  e  $\bar{y} = 1.43264$  aproximações de  $x$  e  $y$ , ambas com seis algarismos corretos.

Neste caso,  $\bar{x} - \bar{y} = 0.00011$  representa o valor exato de  $x - y$  com apenas dois algarismos corretos

A mesma perda de algarismos ocorre se os números tiverem outros expoentes, por exemplo com

$$\bar{x} = 1.43275 * 10^7$$

$$\bar{y} = 1.43264 * 10^7$$

e

$$\bar{x} - \bar{y} = 0.00011 * 10^7$$

Observe-se que  $\bar{x}$  e  $\bar{y}$  são  $O(10^7)$  e  $\bar{x} - \bar{y}$  é  $O(10^3)$ , há perda de 4 algarismos significativos

# O cancelamento subtrativo (2)

---

## Exemplo no Matlab

A fórmula fundamental da trigonometria é  $\sin(x)^2 + \cos(x)^2 = 1$ , mas...

```
>> x=1e-5; 1-cos(x)^2, sin(x)^2  
ans = 1.000000082740371e-10  
ans = 9.999999999666671e-11
```

Qual destes números tem mais algarismos significativos corretos?

É o calculado por  $\sin(x)^2$ , porque ocorre cancelamento subtrativo na diferença  $1 - \cos(x)^2$ .

```
>> cos(1e-5)^2  
ans = 0.9999999999900000
```

```
>> 1-cos(1e-5)^2  
ans = 1.000000082740371e-10
```

## O erro relativo no cancelamento subtrativo

---

```
>> x=1e-5; abs(1-cos(x)^2-sin(x)^2)  
ans = 3.247895369989818e-16
```

O erro absoluto é pequeno mas o erro relativo é muito maior

```
> ans/sin(x)^2
```

```
ans = 3.247895370098080e-06
```

Quanto algarismos significativos corretos tem o valor calculado de  $(1-\cos(x)^2)$ ?

# Propagação do erro relativo

---

Erros relativos grandes podem gerar erros absolutos também grandes

```
>> x=1e-5; (1-cos(x)^2)/(sin(x)^2)
```

```
ans = 0.999996752104630
```

O resultado exato é 1, o erro absoluto é

```
>> abs(1-ans)
```

```
ans = 3.247895370095399e-06
```

# Condicionalismo de uma função

---

O **número de condição de uma função num ponto**  $x$  mede a variação do valor  $f(x)$  provocadas por pequenas alterações (perturbações) no valor do argumento  $x$ .

Exemplo

```
>> x=10; deltaX=1e-5; exp(x+deltaX)-exp(x)
```

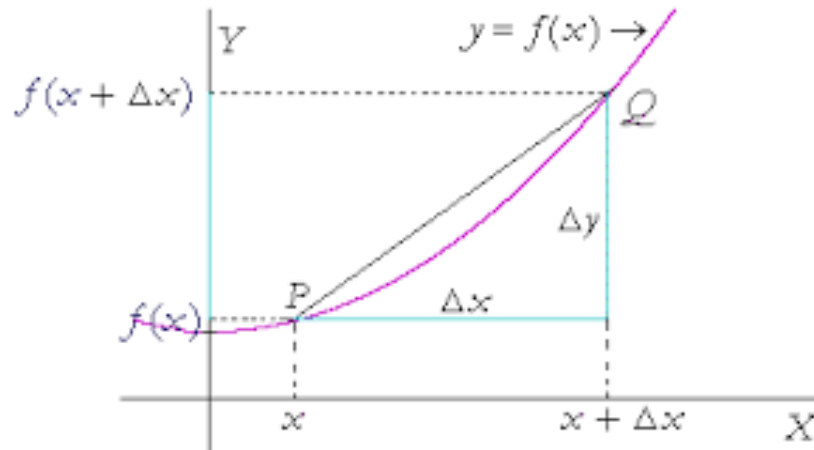
```
ans =
```

```
0.2203
```

Uma perturbação de  $\Delta x = 10^{-5}$  no valor do argumento provocou um erro (absoluto) aproximadamente igual a  $2 * 10^{-1}$ , bastante maior.



# O número de condição absoluto de uma função num ponto



$$\Delta y = |f(x + \Delta x) - f(x)| \approx |f'(x)| \cdot \Delta x$$

**$|f'(x)|$  é o número de condição absoluto de  $f$  no ponto  $x$**

$$f(x + \Delta x) = f(x) + f'(x) \cdot \Delta x + \frac{f''(x)}{2} (\Delta x)^2 + \dots + \frac{f^{(k)}(x)}{k!} (\Delta x)^k + \dots$$

A soma dos termos que se desprezam é o erro de TRUNCATURA

# O número de condição relativo de uma função num ponto

---

De

$$\Delta y = |f(x + \Delta x) - f(x)| \approx |f'(x)| \cdot \Delta x$$

$|f'(x)|$  é o número de condição absoluto de  $f$  no ponto  $x$

com  $x \neq 0$  e  $f(x) \neq 0$ , resulta

$$\left| \frac{\Delta y}{y} \right| = \left| \frac{f(x+\Delta x) - f(x)}{f(x)} \right| \approx \left| \frac{x f'(x)}{f(x)} \right| \cdot \left| \frac{\Delta x}{x} \right|$$

$\left| \frac{x f'(x)}{f(x)} \right|$  **é o número de condição relativo de  $f$  no ponto  $x$**

Exemplo:  $f(x) = e^x, x = 10, \Delta x = 10^{-5}$

$$\Delta y \approx |f'(x)| \cdot \Delta x = e^{10} * 10^{-5} = 0.2203 \dots$$

(erro absoluto cresce  $e^{10} \approx 22026$  vezes)

$$\left| \frac{\Delta y}{y} \right| \approx |x| \cdot \left| \frac{\Delta x}{x} \right| = \mathbf{10} * 10^{-6}$$

(erro relativo cresce 10 vezes)

# Números de condição relativos e perda de algarismos significativos corretos

---

Se o número de condição relativo é da ordem de grandeza de  $10^k$ , então há perda de  $k$  algarismos decimais no valor da função.

## Exemplo 1

format long, x=1; deltaX=1e-5; x+deltaX, exp(x), exp(x+deltaX)

ans =

**1.0000**100000000000

x+deltaX aproxima x com 5 algarismos corretos

ans =

2.718281828459046

número de condição é  $|x|=1$

ans =

**2.7183**09011413245

exp(x+deltaX) aproxima exp(x) com 5 algarismos corretos

## Exemplo 2

x=100; deltaX=1e-3; x+deltaX, exp(x), exp(x+deltaX)

ans =

**1.0000**100000000000e+02

x+deltaX aproxima x com 5 algarismos corretos

ans =

2.688117141816136e+43

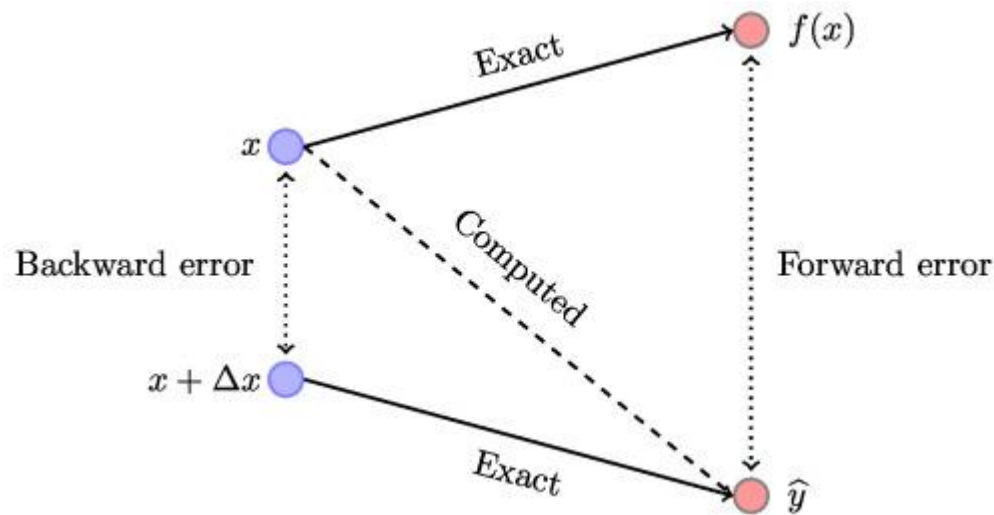
número de condição é  $|x|=100$

ans =

**2.69**0806603464667e+43

exp(x+deltaX) aproxima exp(x) com 3 algarismos corretos

# Erro direto (*forward*) e erro inverso (*backward*)



- O erro direto  $|f(x) - \hat{y}|$  depende do condicionamento de  $f$  no ponto  $x$  e dos erros cometidos no cálculo de  $f(x)$
- O erro inverso  $|\Delta x|$  não depende do condicionamento de  $f$  no ponto  $x$

# Instabilidade numérica *versus* condicionamento

---

Um algoritmo (ou simplesmente uma expressão numérica) diz-se numericamente instável quando para certos dados produzir erros grandes nas operações efetuadas conduzindo a um resultado com pouca (ou nenhuma precisão).

Não confundir instabilidade numérica do algoritmo com condicionamento da função.

Por exemplo, devemos estar atentos à possibilidade de ocorrência de cancelamento subtrativo e substituir uma expressão numérica por outra que seja matematicamente equivalente e que evite o cancelamento.

## **Exemplo**

```
>> 1/(sqrt(10^12+3)-sqrt(10^12))
```

```
ans =
```

```
6.666615903764067e+05
```

multiplicamos numerador e denominador  
por  $\sqrt{10^{12}+3}+\sqrt{10^{12}}$

```
>> (sqrt(10^12+3)+sqrt(10^12))/3
```

```
ans =
```

```
6.6666666666671666e+05
```

# Séries de Taylor (1)

---

Desenvolvimento em série de potências de  $x$  de uma função  $f$  com derivadas contínuas

$$f(x) = f(0) + f'(0)x + \frac{f''(0)}{2}x^2 + \frac{f'''(0)}{3!}x^3 + \dots + \frac{f^{(k)}(0)}{k!}x^k + \dots$$

Exemplos

$$\begin{aligned}\sin(x) &= \sin(0) + \cos(0)x - \frac{\sin(0)}{2}x^2 - \frac{\cos(0)}{3!}x^3 + \frac{\sin(0)}{4!}x^4 + \frac{\cos(0)}{5!}x^5 + \dots \\ &= x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \dots\end{aligned}$$

$$\begin{aligned}\cos(x) &= \cos(0) - \sin(0)x - \frac{\cos(0)}{2}x^2 + \frac{\sin(0)}{3!}x^3 + \frac{\cos(0)}{4!}x^4 - \frac{\sin(0)}{5!}x^5 + \dots \\ &= 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \dots\end{aligned}$$

$$\begin{aligned}\exp(x) &= \exp(0) + \exp(0)x + \frac{\exp(0)}{2}x^2 + \frac{\exp(0)}{3!}x^3 + \frac{\exp(0)}{4!}x^4 + \frac{\exp(0)}{5!}x^5 + \dots \\ &= 1 + x + \frac{x^2}{2} + \frac{x^3}{3!} + \frac{x^4}{4!} + \frac{x^5}{5!} + \dots\end{aligned}$$

# Séries de Taylor (2)

Desenvolvimento em série de potências de  $x-a$  de uma função  $f$  com derivadas contínuas

$$f(x) = f(a) + f'(a)(x-a) + \frac{f''(a)}{2}(x-a)^2 + \frac{f'''(a)}{3!}(x-a)^3 + \dots + \frac{f^{(k)}(a)}{k!}(x-a)^k + \dots$$

Exemplo 1 ( $f(x) = \sin(x)$ ,  $a = \pi/2$ )

$$\begin{aligned} \sin(x) &= \sin(\pi/2) + \cos(\pi/2)(x - \pi/2) - \frac{\sin(\pi/2)}{2}(x - \pi/2)^2 - \frac{\cos(\pi/2)}{3!}(x - \pi/2)^3 + \frac{\sin(\pi/2)}{4!}(x - \pi/2)^4 + \dots \\ &= 1 - \frac{(x-\pi/2)^2}{2!} + \frac{(x-\pi/2)^4}{4!} - \frac{(x-\pi/2)^6}{6!} \dots \end{aligned}$$

Exemplo 2 ( $f(x) = \log(x)$ ,  $a = 1$ )

Para o logaritmo natural (de base e)  $f(x) = \log(x)$ , tem-se  $f'(x) = x^{-1}$ ,  $f''(x) = -x^{-2}$ ,  $f'''(x) = 2x^{-3}$ ,  $f^{(iv)}(x) = -3 * 2x^{-4}$ ,  $f^{(v)}(x) = 4 * 3 * 2x^{-5}$ , ...  $f^{(k)}(1) = (-1)^{k+1}(k-1)!$

$$\log(x) = (x-1) - \frac{(x-1)^2}{2} + \frac{(x-1)^3}{3} + \dots + (-1)^{k+1} \frac{(x-1)^k}{k} + \dots \text{ válida para } 0 < x \leq 2$$

$$\log(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} + \dots + (-1)^{k+1} \frac{x^k}{k} + \dots \text{ válida para } -1 < x \leq 1$$

# Majoração dos erros de truncatura (1)

---

$$\begin{aligned} f(x) &= f(a) + f'(a)(x-a) + \frac{f''(a)}{2}(x-a)^2 + \frac{f'''(a)}{3!}(x-a)^3 + \dots + \frac{f^{(k)}(a)}{k!}(x-a)^k + R_k(x) \\ &= T_k(x) + R_k(x) \end{aligned}$$

$T_k(x)$  aproxima o valor de  $f(x)$  com erro de truncatura  $R_k(x)$ .

Resto na forma de Lagrange

$$R_k(x) = \frac{f^{(k+1)}(\theta)}{(k+1)!} (x-a)^{k+1}$$

$\theta$  é um ponto que está entre  $a$  e  $x$ .

Se  $|f^{(k+1)}(\theta)| \leq M$ , para qualquer  $\theta$  entre  $a$  e  $x$ , então

$$|R_k(x)| \leq \frac{M}{(k+1)!} |x-a|^{k+1}$$



# Majoração dos erros de truncatura (2)

---

Exemplo

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} + R_6$$

com  $R_6(x) = \frac{f^{(7)}(\theta)}{7!} x^7$ .

Para  $x = \frac{\pi}{6}$ , no Matlab

```
>> x=pi/6; T6=x-x^3/factorial(3)+x^5/factorial(5)
```

```
T6 = 0.500002132588792
```

e  $|R_6\left(\frac{\pi}{6}\right)| \leq \frac{1}{7!} \left(\frac{\pi}{6}\right)^7$

```
>> (pi/6)^7/factorial(7)
```

```
ans = 2.140719769235796e-06
```

Nota: o valor exato é 0.5, o erro é 2.132588792e-06

# Majoração dos erros de truncatura (3)

---

Numa série alternada, o erro de truncatura é inferior ao valor absoluto do primeiro termo que se despreza

$$\exp(x) = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \frac{x^5}{5!} + \dots$$

com  $x = -1$ ,

$T_4(-1) = 1 - 1 + \frac{1}{2} - \frac{1}{3!} + \frac{1}{4!}$  aproxima o valor de  $\exp(-1)$  com erro de truncatura inferior a  $\frac{1}{5!}$

```
>> T4=1/2-1/factorial(3)+1/factorial(4)
```

```
T4 =0.3750000000000000
```

```
>> abs(exp(-1)-T4)
```

```
ans =0.007120558828558
```

```
>> 1/factorial(5)
```

```
ans =0.0083333333333333
```

## II. Resolução de equações não-lineares

---

Determinar  $x$  tal que

$$f(x) = 0$$

(raízes da equação ou zeros de  $f$ )

Fórmula resolvente da equação polinomial de segundo grau  $ax^2 + bx + c = 0$

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Como calcular as raízes da equação  $x^5 - x^4 + 2x^3 - 3x^2 - 5x + 6 = 0$  ?

No Matlab

```
>> roots([1,-1,2,-3,-5,6])
```

```
ans =
```

```
-0.0943 + 1.9135i
```

```
-0.0943 - 1.9135i
```

```
-1.1878 + 0.0000i
```

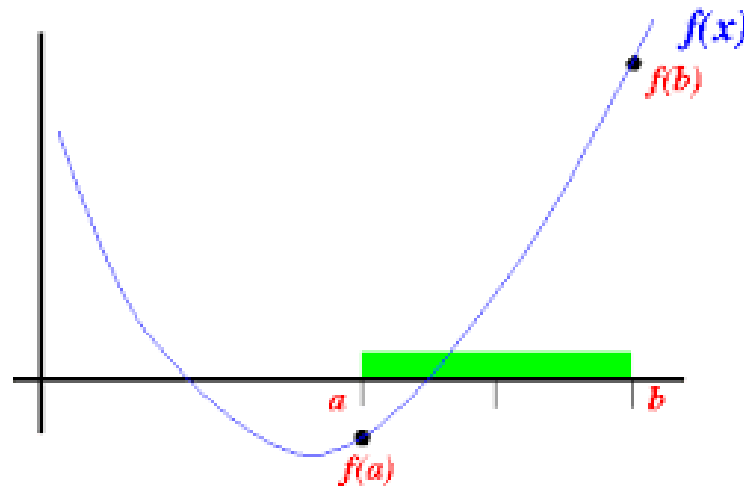
```
1.3763 + 0.0000i
```

```
1.0000 + 0.0000i
```

# A importância da continuidade da função $f$

---

Se  $f$  é contínua  $[a, b]$  e  $f(a) \cdot f(b) < 0$ , então existe pelo menos uma raiz  $r$  da equação  $f(x) = 0$  entre  $a$  e  $b$ .



# Método da bisseção

---

$f$  é contínua  $[a, b]$  e  $f(a) \cdot f(b) < 0$

Em cada iteração, divide-se em duas partes iguais e escolhe-se o subintervalo em que  $f$  muda de sinal:

Calcula-se o valor de  $f$  no ponto médio  $m = (a + b)/2$

Se  $f(m) = 0$  então  $r \leftarrow m$

Se  $f(m) \cdot f(a) < 0$  (a raiz está entre  $a$  e  $m$ ) faz-se  $b \leftarrow m$ .

Se  $f(m) \cdot f(a) > 0$  (a raiz está entre  $m$  e  $b$ ) faz-se  $a \leftarrow m$ .

(ver figura 2.2 na p.42 do livro)

# Método da bisseção. Exemplo

---

(ver Exemplo 2.1 do livro) No início de cada ano o cliente de um banco deposita  $v$  euros num fundo de investimento e retira ao fim do  $n$ -ésimo ano um capital de  $M$  euros. Queremos calcular a taxa de juro anual  $r$  deste investimento.

Dado que

$$M = v \sum_{k=1}^n (1 + r)^k = v \frac{1+r}{r} [(1+r)^n - 1],$$

$r$  é uma raiz da equação  $f(x) = 0$ , onde

$$f(x) = M - v \frac{1+x}{x} [(1+x)^n - 1].$$

Consideremos que o investidor deposita anualmente  $v = 1000$  e que depois de 5 anos recebe o capital  $M = 6000$  euros. Qual a taxa de juro anual que lhe pagou o banco?

```
>> f=inline('6000-1000*(1+x)/x*((1+x)^5-1)'); fplot(f,[0.01,0.3])
```

# Método da bisseção. Exemplo (continuação)

---

Vemos que  $f$  tem um zero entre  $a = 0.01$  e  $b = 0.3$ .

No Matlab,

```
>> a=0.01; b=0.3; f(a), f(b)
```

```
ans = 847.9849
```

```
ans = -5.7560e+03
```

Primeira iteração

```
>> m=(a+b)/2, f(m)
```

```
m = 0.1550
```

```
ans = -1.8649e+03
```

Porque  $f(a)$  e  $f(m)$  têm sinais contrários, a raiz está entre  $a=0.01$  e  $m=0.155$ . Fazemos  $b=m$  e continuamos a iterar com o novo intervalo  $[a,b]=[0.01,0.155]$  cuja amplitude é metade da amplitude do intervalo inicial  $[0.01,0.3]$

# A convergência do método da bisseção (1)

---

Ao fim de  $k$  iterações, temos o intervalo  $[a^{(k)}, b^{(k)}]$  tal que

$$b^{(k)} - a^{(k)} = \frac{b-a}{2^k}$$

O erro na aproximação

$$x^{(k)} = (a^{(k)} + b^{(k)})/2$$

$$\text{é } |e^{(k)}| = |x^{(k)} - r| < \frac{b-a}{2^{k+1}}$$

Para garantir que  $|e^{(k)}| < \text{tol}$ , basta fazer  $k_{\min}$  iterações onde  $k_{\min}$  é o menor inteiro positivo que satisfaz a desigualdade

$$k_{\min} > \log_2 \left( \frac{b-a}{\text{tol}} \right) - 1$$

$$\text{c.a.: } \frac{b-a}{2^{k+1}} < \text{tol} \Leftrightarrow \frac{b-a}{\text{tol}} < 2^{k+1} \Leftrightarrow \log_2 \left( \frac{b-a}{\text{tol}} \right) < k + 1$$



# A convergência do método da bisseção (2)

---

Voltando ao problema anterior da taxa de juro, com  $a = 0.01$ ,  $b = 0.3$  e  $tol = 1e - 4$ , de

```
>> log2((b-a)/1e-5)-1  
ans =  
    13.8238
```

concluimos que  $k_{min}=14$

nota: se efetuarmos 14 iterações obtemos a aproximação

$$x^{(14)}=0.0614...$$

que corresponde a uma taxa de juro anual de 6,14%

# Vantagens e desvantagens do método da bisseção

---

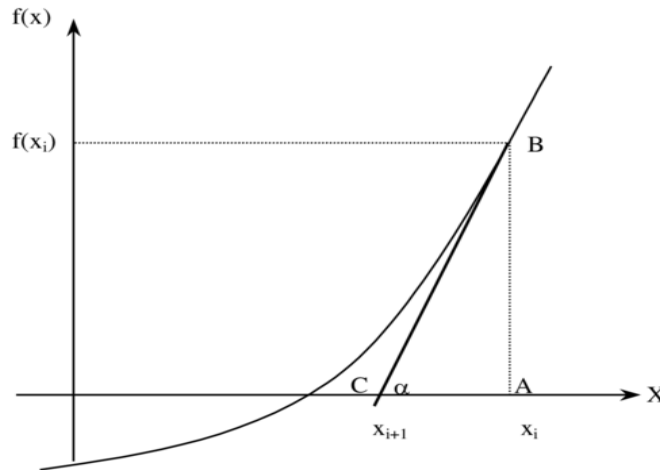
## VANTAGENS:

- Convergência garantida
- A função  $f$  não precisa de ser derivável (basta que seja contínua)
- Cada iteração requer apenas o cálculo da função num ponto (o ponto médio)
- Excelente estabilidade numérica (erros no cálculo de  $f$  não afetam o resultado desde que o sinal do valor calculado esteja correto)

## DESVANTAGEM (única):

- Convergência lenta (linear); cada iteração acrescenta apenas mais um bit correto à aproximação

# O método de Newton-Raphson (das tangentes)



$$\tan(\alpha) = \frac{AB}{AC}$$

$$f'(x_i) = \frac{f(x_i)}{x_i - x_{i+1}}$$

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

$x^{(i+1)}$  é a abcissa do ponto em que a reta tangente à curva no ponto de abcissa  $x^{(i)}$  interseja o eixo dos xx

[https://upload.wikimedia.org/wikipedia/commons/e/e0/NewtonIteration\\_Ani.gif](https://upload.wikimedia.org/wikipedia/commons/e/e0/NewtonIteration_Ani.gif)

# A evolução do erro de truncatura

---

Do desenvolvimento em série de Taylor na p. 42, com  $x = r$  e  $a = x^{(i)}$ :

$$f(r) = f(x^{(i)}) + f'(x^{(i)}) (r - x^{(i)}) + \frac{f''(\theta)}{2} (r - x^{(i)})^2$$

e assumindo que  $f'(x^{(i)}) \neq 0$ , resulta

$$r - x^{(i)} = -\frac{f(x^{(i)})}{f'(x^{(i)})} - \frac{f''(\theta)}{2f'(x^{(i)})} (r - x^{(i)})^2$$

$$r = x^{(i)} - \frac{f(x^{(i)})}{f'(x^{(i)})} - \frac{f''(\theta)}{2f'(x^{(i)})} (r - x^{(i)})^2$$

e

$$r - x^{(i+1)} = -\frac{f''(\theta)}{2f'(x^{(i)})} (r - x^{(i)})^2$$

# A convergência quadrática

---

Resulta

$$r - x^{(i+1)} = - \frac{f''(\theta)}{2f'(x^{(i)})} (r - x^{(i)})^2$$

Conclusão:  $x^{(i+1)} = x^{(i)} - \frac{f(x^{(i)})}{f'(x^{(i)})}$  aproxima o valor da raiz  $r$  com erro de truncatura proporcional ao quadrado do erro da aproximação  $x^{(i)}$ .

Por exemplo, Se  $|r - x^{(i)}| \approx 10^{-3}$  então

$$r - x^{(i+1)} \approx - \frac{f''(\theta)}{2f'(x^{(i)})} (10^{-3})^2$$

e se  $|\frac{f''(\theta)}{2f'(x^{(i)})}| \approx 1$  então  $|r - x^{(i+1)}| \approx 10^{-6}$ .

Para a iteração seguinte

$$r - x^{(i+2)} = - \frac{f''(\mu)}{2f'(x^{(i+1)})} (r - x^{(i+1)})^2$$

onde  $\mu$  está entre  $r$  e  $x^{(i+1)}$  ...

# Ordem de convergência

---

Se  $f'(r) \neq 0$ ,

$$\lim_{i \rightarrow +\infty} \frac{r - x^{(i+1)}}{(r - x^{(i)})^2} = - \frac{f''(r)}{2f'(r)}$$

Definição: num método iterativo, se

$$\lim_{i \rightarrow +\infty} \frac{|r - x^{(i+1)}|}{|r - x^{(i)}|^p} = C > 0$$

- ▣ para  $i$  suficientemente grande,  $r - x^{(i+1)} \approx C \cdot (r - x^{(i)})^p$
- ▣  $p$  (positivo não necessariamente inteiro) é a ordem de convergência do método
- ▣  $C$  é a constante de convergência assintótica
- ▣  $p = 2$  no caso do método de newton-Raphson (convergência quadrática)
- ▣  $p = 1$  no caso do método da bisseção (convergência linear)

# Exemplo no Matlab

---

```
>> df=inline('-1000*(-1/x^2*((1+x)^5-1)+(1+1/x)*5*(1+x)^4)')
```

```
>> x= 0.3; % aproximação inicial
```

```
>> x=x-f(x)/df(x) % 5ª iteração  
x=0.061402411536525
```

```
>> x=x-f(x)/df(x) % 1ª iteração  
x=0.118642027821101
```

```
>> x=x-f(x)/df(x) % 6ª iteração  
x=0.061402411536525
```

```
>> x=x-f(x)/df(x) % 2ª iteração  
x=0.065390200813148
```

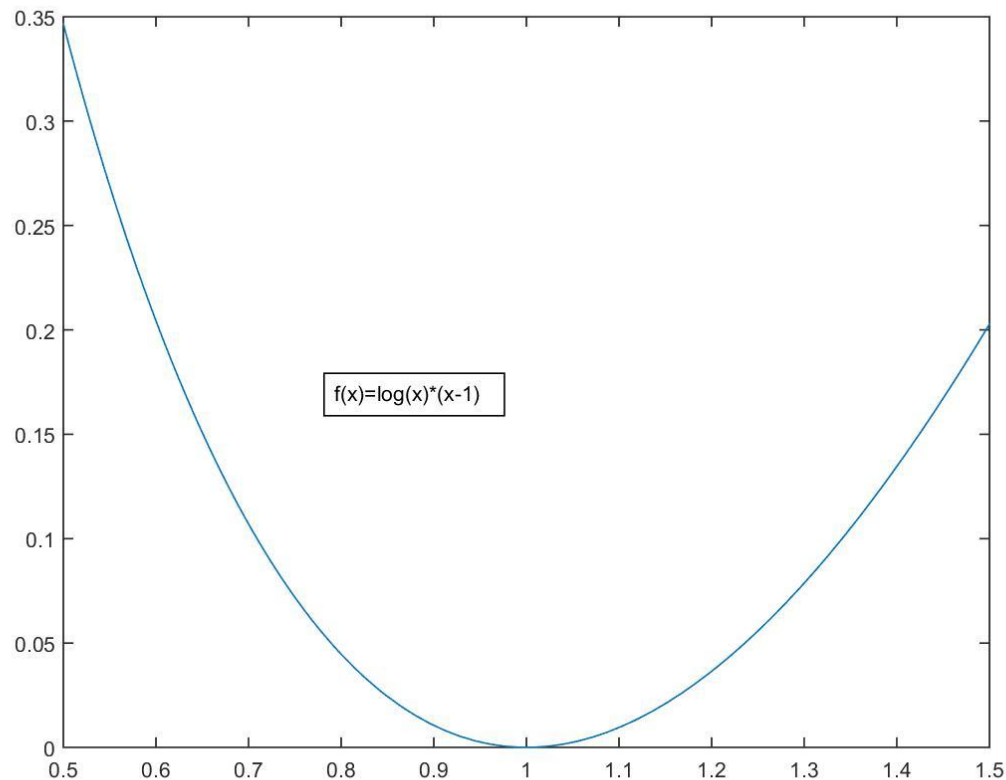
NOTA: o método da bissecção precisou de 14 iterações para produzir 0.0614...

```
>> x=x-f(x)/df(x) % 3ª iteração  
x=0.061422972148339
```

```
>> x=x-f(x)/df(x) % 4ª iteração  
x=0.061402412085601
```

# Quando a raiz não é simples (1)

Se  $r$  tem multiplicidade superior a 1, é  $f'(r) = 0$  (à medida que  $x^{(i)}$  se aproxima da raiz, a reta tangente tende para o eixo dos  $xx$ ). A convergência é apenas linear.





# Um exemplo de aplicação

---

Nos primeiros modelos de computadores digitais a divisão não era efetuada por hardware mas sim por software. Assim, a divisão de  $a$  por  $b$ , implicava a multiplicação de  $a$  pelo inverso de  $b$ .

O inverso aritmético de um número  $b \neq 0$  é a raiz da equação  $b - \frac{1}{x} = 0$ .

A fórmula iterativa  $x^{(i+1)} = x^{(i)} - \frac{f(x^{(i)})}{f'(x^{(i)})}$

dá

$$x^{(i+1)} = x^{(i)} - \frac{b - \frac{1}{x^{(i)}}}{\frac{1}{(x^{(i)})^2}} = x^{(i)} - (b(x^{(i)})^2 - x^{(i)})$$

ou seja  $x^{(i+1)} = x^{(i)} (2 - bx^{(i)})$

Para calcular o valor de  $1/7$  (sem usar divisão), podemos começar com  $x^{(0)} = 0.1$  e usar a fórmula anterior para calcular  $x^{(1)}, x^{(2)}, \dots$

# O método de Newton nem sempre converge

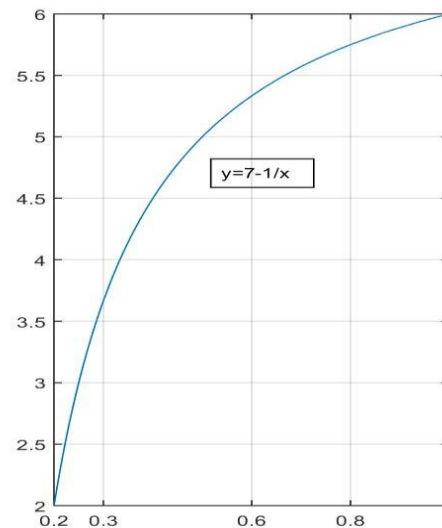
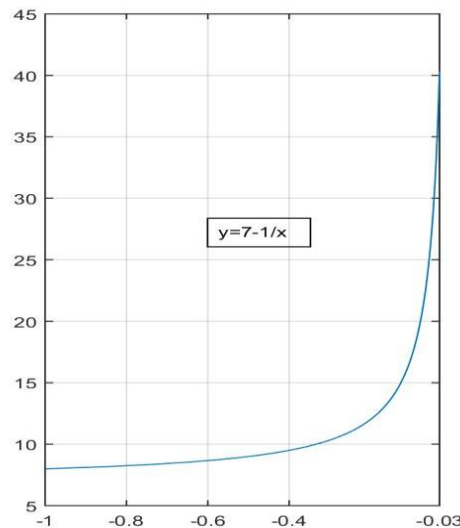
No exemplo anterior, começando com  $x^{(0)}=0.3$ , o método diverge.

A tangente à curva no ponto de abscissa 0.3 intersesta o eixo dos xx no ponto de abscissa -0.03

```
>> b=7; x=0.3;
```

```
>> x=x*(2-b*x)
```

```
x = -0.0300000000000000
```



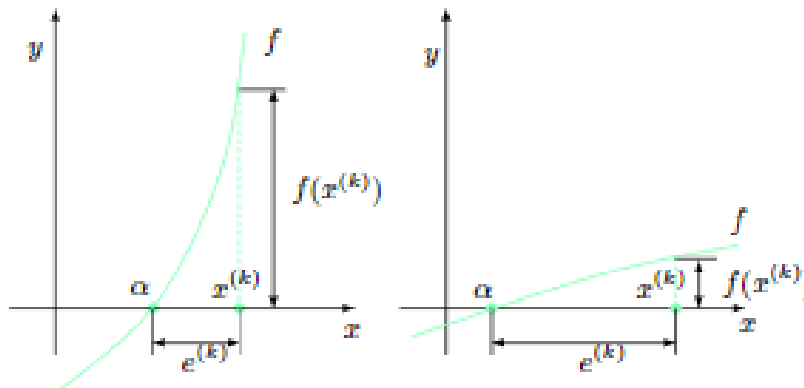
# Critérios de paragem

$x^{(0)}, x^{(1)}, x^{(2)}, \dots, x^{(k)}, \dots \rightarrow \alpha$       parar quando  $|e^{(k)}| = |x^{(k)} - \alpha| < tol$  ?

Fixada uma tolerância  $tol$ :

- $|x^{(k)} - x^{(k-1)}| < tol$  (teste sobre o incremento)
- $|f(x^{(k)})| < tol$  (teste sobre o resíduo)

O erro  $|x^{(k)} - \alpha|$  e o resíduo  $|f(x^{(k)})|$  podem ser muito diferentes



# O método do ponto fixo (1)

---

No Matlab, partindo de um qualquer valor real  $x$ , e repetindo o comando

```
>> x=cos(x)
```

a sucessão de valores converge (embora lentamente) para 0.739085133215161

Este número diz-se um **ponto fixo** da função coseno. É uma raiz da equação  $x - \cos(x) = 0$ .

Em geral, um ponto fixo de uma função  $\varphi$  é uma raiz da equação

$$x = \varphi(x)$$

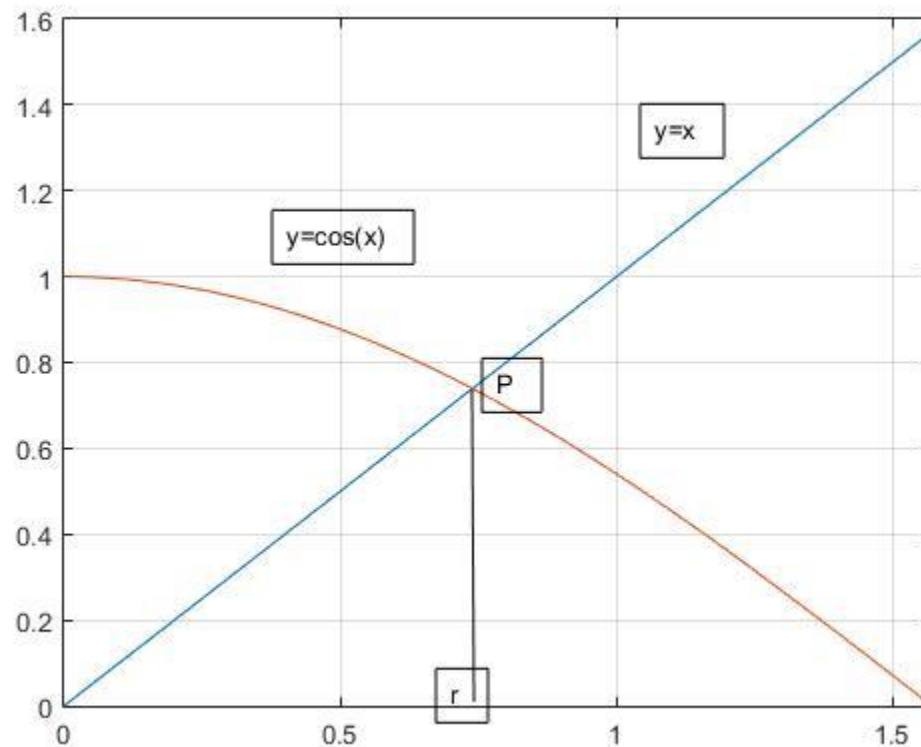
ou

$$f(x) = 0$$

com  $f(x) = x - \varphi(x)$ .

## O método do ponto fixo (2)

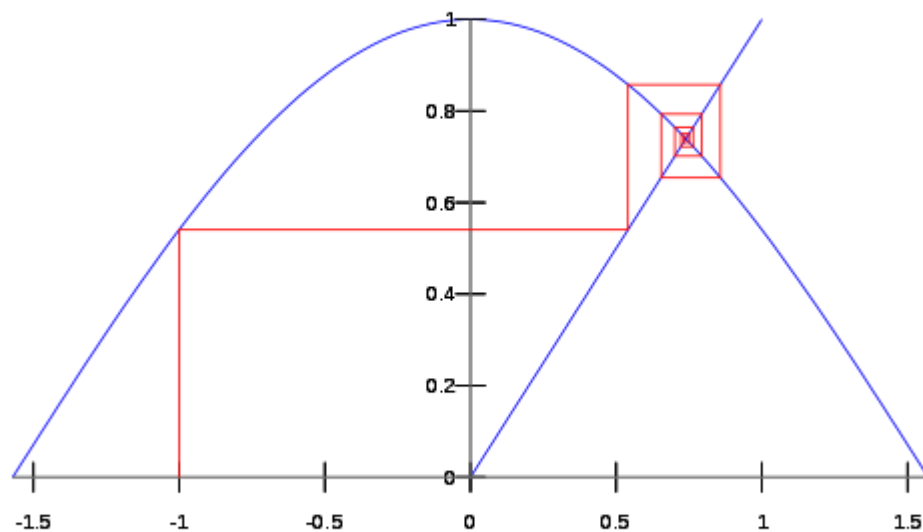
O ponto fixo de  $\cos(x)$  é a abscissa do ponto P em que a curva da função cosseno intersecta a reta  $y=x$ .



# Interpretação geométrica das iterações do ponto fixo

O ponto fixo de  $\cos(x)$  é a abscissa do ponto P em que a curva da função coseno intersesta a reta  $y=x$ .

$x^{(0)} = -1$  aproximação inicial,  $x^{(1)} = \cos(-1) = 0.54 \dots$ ,  $x^{(2)} = \cos(0.54 \dots) = 0.85 \dots$



# Análise da convergência do método do ponto fixo (1)

---

$$x = \varphi(x)$$

A partir da aproximação inicial  $x^{(0)}$ ,  $x^{(1)} = \varphi(x^{(0)})$ ,  $x^{(2)} = \varphi(x^{(1)})$ , ...  $x^{(k+1)} = \varphi(x^{(k)})$

Que condições devem satisfazer a função iteradora  $\varphi$  e o valor inicial  $x^{(0)}$  para que a sucessão seja convergente para o ponto fixo?

**Teorema do valor médio de Lagrange:** se  $\varphi$  tem derivada contínua em  $[a,b]$ , então existe pelo menos um ponto  $\theta$  entre  $a$  e  $b$  tal que

$$\varphi'(\theta) = \frac{\varphi(b) - \varphi(a)}{b - a}$$

Interpretação intuitiva: se  $\varphi(b) - \varphi(a)$  representar a distância percorrida desde o instante  $a$  até ao instante  $b$ , então  $\frac{\varphi(b) - \varphi(a)}{b - a}$  é a velocidade média nesse intervalo de tempo. O valor  $\varphi'(\theta)$  é a velocidade (instantânea) no instante  $\theta$ . Em pelo menos um instante, a velocidade é igual à velocidade média.

Aplicando este resultado à função iteradora  $\varphi$ :  $x = \varphi(x)$

# Análise da convergência do método do ponto fixo (2)

---

Aplicando o teorema do valor médio à **função iteradora**  $\varphi$  no intervalo  $[x^{(k)}, r]$ : existe  $\theta$  entre  $x^{(k)}$  e  $r$  tal que

$$\varphi'(\theta) = \frac{\varphi(x^{(k)}) - \varphi(r)}{x^{(k)} - r}$$

resulta  $\varphi(x^{(k)}) - \varphi(r) = \varphi'(\theta)(x^{(k)} - r)$ .

Uma vez que  $\varphi(x^{(k)}) = x^{(k+1)}$  e  $\varphi(r) = r$ :

$$x^{(k+1)} - r = \varphi'(\theta)(x^{(k)} - r)$$

O erro na iteração  $k+1$  é igual ao erro na iteração anterior multiplicado por  $\varphi'(\theta)$ . Se  $|\varphi'(\theta)| < 1$  então  $|x^{(k+1)} - r| < |x^{(k)} - r|$ .

Se existir  $M \in ]0, 1[$  tal que  $|\varphi'(x)| < M$  num intervalo  $I$  centrado em  $r$  e  $x^{(0)} \in I$ , então o método converge porque

$$|x^{(k)} - r| < M^k |x^{(0)} - r| \text{ e } M^k \rightarrow 0$$



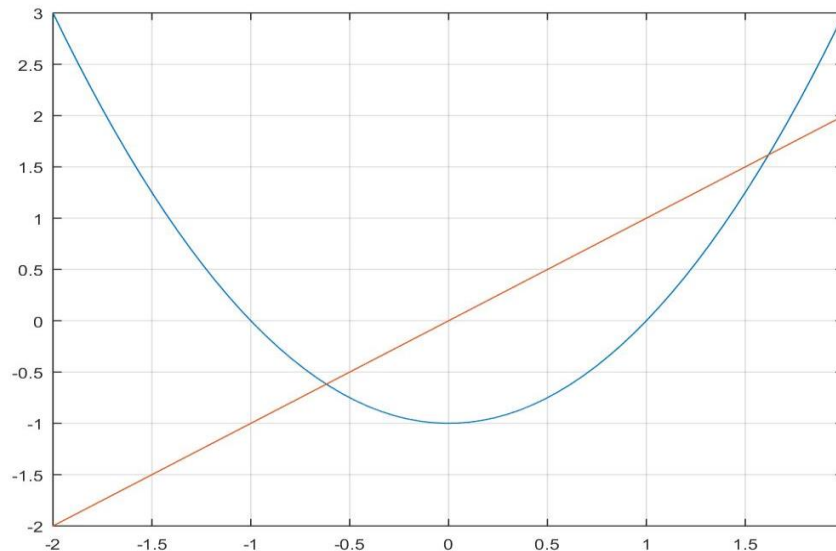
# Análise da convergência do método do ponto fixo (3)

**Exemplo 1:** a função  $\varphi(x) = \cos(x)$  satisfaz a condição requerida:

$$\varphi'(r) = \sin(r) = \sin(0.7391 \dots) \approx 0.67$$

Em qualquer intervalo  $I = [r - a, r + a]$  que não contenha  $\pi/2$  tem-se  $|\varphi'(x)| < M < 1$  e a convergência está garantida com  $x^{(0)} \in I$ .

**Exemplo 2:**  $\varphi(x) = x^2 - 1$  tem dois pontos fixos  $r_{\pm} = \frac{1 \pm \sqrt{5}}{2}$  mas  $|\varphi'(r_{\pm})| = |1 \pm \sqrt{5}| > 1$ . Não há convergência.



# A escolha da função iteradora

(1)

Dada a equação  $f(x) = 0$  há infinitas maneiras de a reescrever na forma  $x = \varphi(x)$ , por exemplo  $x = x + f(x)$ . As boas escolhas são aquelas que cumprem a condição de ser  $|\varphi'(x)|$  próximo de zero numa vizinhança da raiz da equação.

**Exemplo** Com  $x \neq 0$ , a equação  $\frac{1}{x} - e^x = 0$  pode escrever-se na forma

i.  $x = e^{-x}, \quad \varphi_1(x) = e^{-x} \Rightarrow \varphi'_1(x) = -e^{-x}$

ii.  $x = -\log(x), \quad \varphi_2(x) = -\log(x) \Rightarrow \varphi'_2(x) = -\frac{1}{x}$

iii.  $x = \frac{x+e^{-x}}{2}, \quad \varphi_3(x) = \frac{x+e^{-x}}{2}, \Rightarrow \varphi'_3(x) = \frac{1-e^{-x}}{2}$

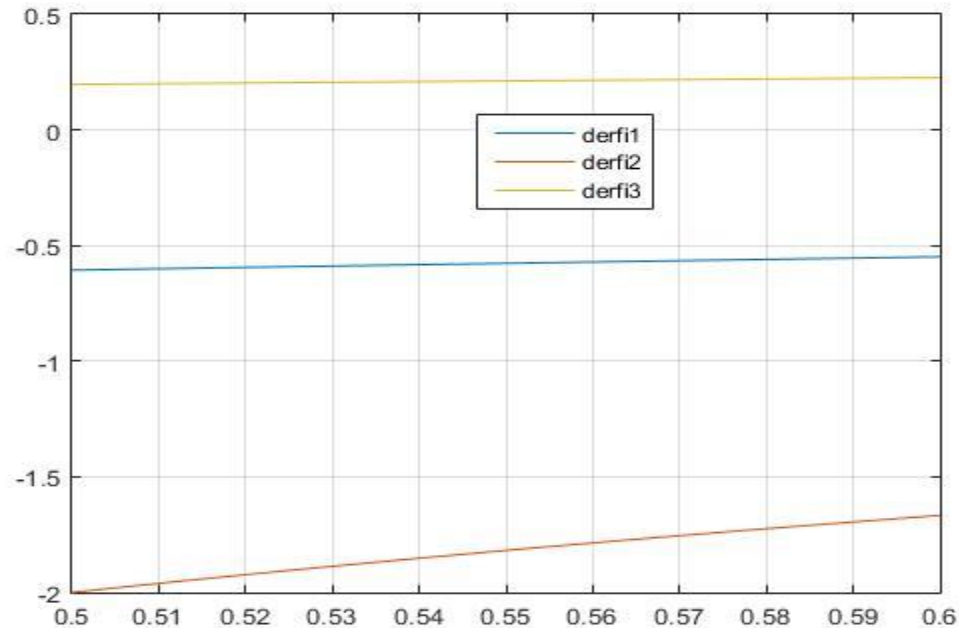
Começando por observar que a raiz da equação está entre 0.5 e 0.6, no Matlab produzimos os gráficos de  $\varphi'_1, \varphi'_2$  e  $\varphi'_3$  no intervalo  $[0.5, 0.6]$ :

```
>> fplot('[-exp(-x), -1/x, (1-exp(-x))/2]', [0.5, 0.6])
```

```
>> grid on, legend('derfi1','derfi2','derfi3')
```

# A escolha da função iteradora

(2)



Conclusões:

- As funções iteradoras  $\varphi_1$  e  $\varphi_3$  produzem sucessões convergentes desde que  $0.5 < x^{(0)} < 0.6$  porque neste intervalo é  $|\varphi'_1(x)| < 1$  e é  $|\varphi'_3(x)| < 1$ ;
- $\varphi_3$  produz convergência mais rápida porque  $|\varphi'_3(x)| < |\varphi'_1(x)|$ ;
- A sequência produzida com  $\varphi_2$  diverge porque  $|\varphi'_2(x)| > 1$

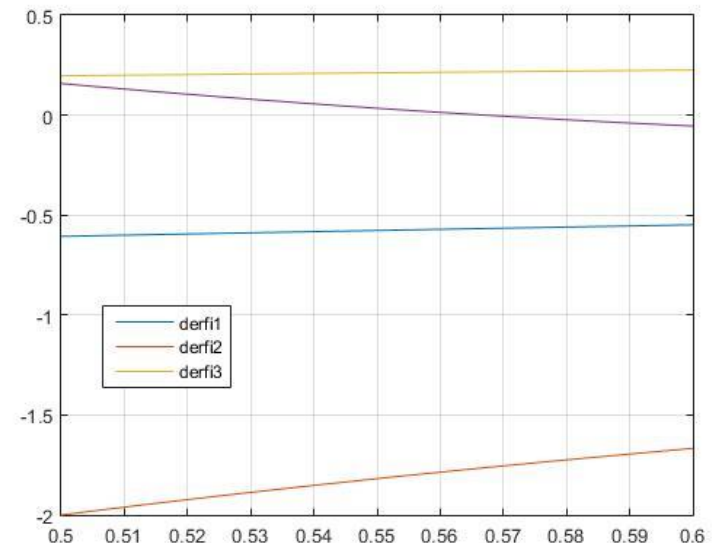
# Uma escolha especial da função iteradora

No método de Newton-Raphson, a equação  $f(x) = 0$  é reescrita na forma

$$x = x - \frac{f(x)}{f'(x)} \text{ ou seja } x = \varphi(x), \text{ com } \varphi(x) = x - \frac{f(x)}{f'(x)}.$$

$$\text{Tem-se } \varphi'(x) = 1 - \frac{(f'(x))^2 - f(x)f''(x)}{(f'(x))^2} = \frac{f(x)f''(x)}{(f'(x))^2} \text{ e } \varphi'(r) = 0.$$

Na figura anterior, vamos sobrepor o gráfico de  $\varphi'(x)$  para esta escolha de  $\varphi$  :



```
>> hold on, fplot('(1/x-exp(x))*(2/x^3-exp(x))/(-1/x^2-exp(x))^2',[0.5,0.6])
```

# Estimativa do erro $|x^{(k+1)} - r|$ (1)

---

Se pararmos as iterações quando  $|x^{(k+1)} - x^{(k)}| < \text{tol}$ , podemos garantir  $|r - x^{(k)}| < \text{tol}$  ?  
Em geral, não. Tal depende dos valores de  $|\varphi'(x)|$  na vizinhança da raiz  $r$ .

$$r - x^{(k)} = (r - x^{(k+1)}) + (x^{(k+1)} - x^{(k)})$$

e, uma vez que

$$r - x^{(k+1)} = \varphi'(\theta)(r - x^{(k)})$$

resulta

$$r - x^{(k)} = \varphi'(\theta)(r - x^{(k)}) + (x^{(k+1)} - x^{(k)})$$

ou seja

$$r - x^{(k)} = \frac{1}{1 - \varphi'(\theta)} (x^{(k+1)} - x^{(k)})$$

- ▣ Se  $\varphi'(x) \approx 0$  numa vizinhança de  $r$ , então a diferença entre duas iteradas sucessivas dá uma boa estimativa do erro.
- ▣  $-1 < \varphi'(x) < 0 \Rightarrow |r - x^{(k)}| < |x^{(k+1)} - x^{(k)}|$
- ▣  $\varphi'(x) \approx 1 \Rightarrow |r - x^{(k)}| \gg |x^{(k+1)} - x^{(k)}|$

## Estimativa do erro $|x^{(k+1)} - r|$ (2)

Exemplo: a equação  $(x - 1)^3 = 0$  tem a raiz  $r=1$  e a fórmula iterativa  $x^{(i+1)} = x^{(i)} - \frac{f(x^{(i)})}{f'(x^{(i)})}$  com  $f(x) = (x - 1)^3$  dá

$$x^{(i+1)} = x^{(i)} - \frac{x^{(i)} - 1}{3}$$

Tem-se  $x^{(i+1)} - r = \frac{2}{3}(x^{(i)} - r)$  e a convergência é linear.

A função iteradora é  $\varphi(x) = x - \frac{x-1}{3}$  e a derivada é  $\varphi'(x) = \frac{2}{3}$  (constante)

Resulta

$$r - x^{(k)} = \frac{1}{1 - \frac{2}{3}} (x^{(k+1)} - x^{(k)}) = 3 (x^{(k+1)} - x^{(k)})$$

# O condicionamento das raízes (1)

No Matlab vamos calcular os coeficientes do polinómio mónico que tem os zeros 1, 1.999, 2 e 2.001.

```
>> p=poly([1, 1.999, 2, 2.001])  
p =  
    1.0000 -7.0000 18.0000 -20.0000  8.0000,
```

Trata-se do polinómio  $p(x) \approx x^4 - 7x^3 + 18x^2 - 20x + 8$

Vamos introduzir uma perturbação igual a  $10^{-4}$  num dos coeficientes, por exemplo, vamos considerar o polinómio perturbado  $\tilde{p}(x) \approx 1.0001x^4 - 7x^3 + 18x^2 - 20x + 8$ .

Qual será o efeito desta perturbação sobre os valores dos zeros 1, 1.999, 2 e 2.001 ? Da ordem de grandeza da perturbação  $10^{-4}$  ou maior?

```
>> p(1)=p(1)+1e-4; r=roots(p)  
r =  
    2.0559 + 0.1052i  
    2.0559 - 0.1052i  
    1.8873 + 0.0000i  
    1.0001 + 0.0000i
```

# O condicionamento das raízes

(2)

O erro é maior do que a perturbação  $10^{-4}$  no caso dos zeros 1.999, 2 e 2.001 mas não no caso do zero igual a 1:

```
>> err=abs(r-[2.001; 2; 1.999; 1])
```

err =

0.1187

0.1192

0.1117

0.0001

Como se explica isto? Pode mostrar-se que o número de condição (absoluto) de uma raiz  $r$  da equação  $f(x) = 0$  é igual a  $\frac{1}{|f'(r)|}$ .

$$p(x) = x^4 - 7x^3 + 18x^2 - 20x + 8$$

$$p'(x) = 4x^3 - 21x^2 + 36x - 20$$

Tem-se  $p'(2.001) = 3.004e - 6$  e o erro 0.1187 é bem maior do que  $10^{-4}$  (embora não tão grande quanto  $\frac{10^{-4}}{3 \cdot 10^{-6}}$ ).

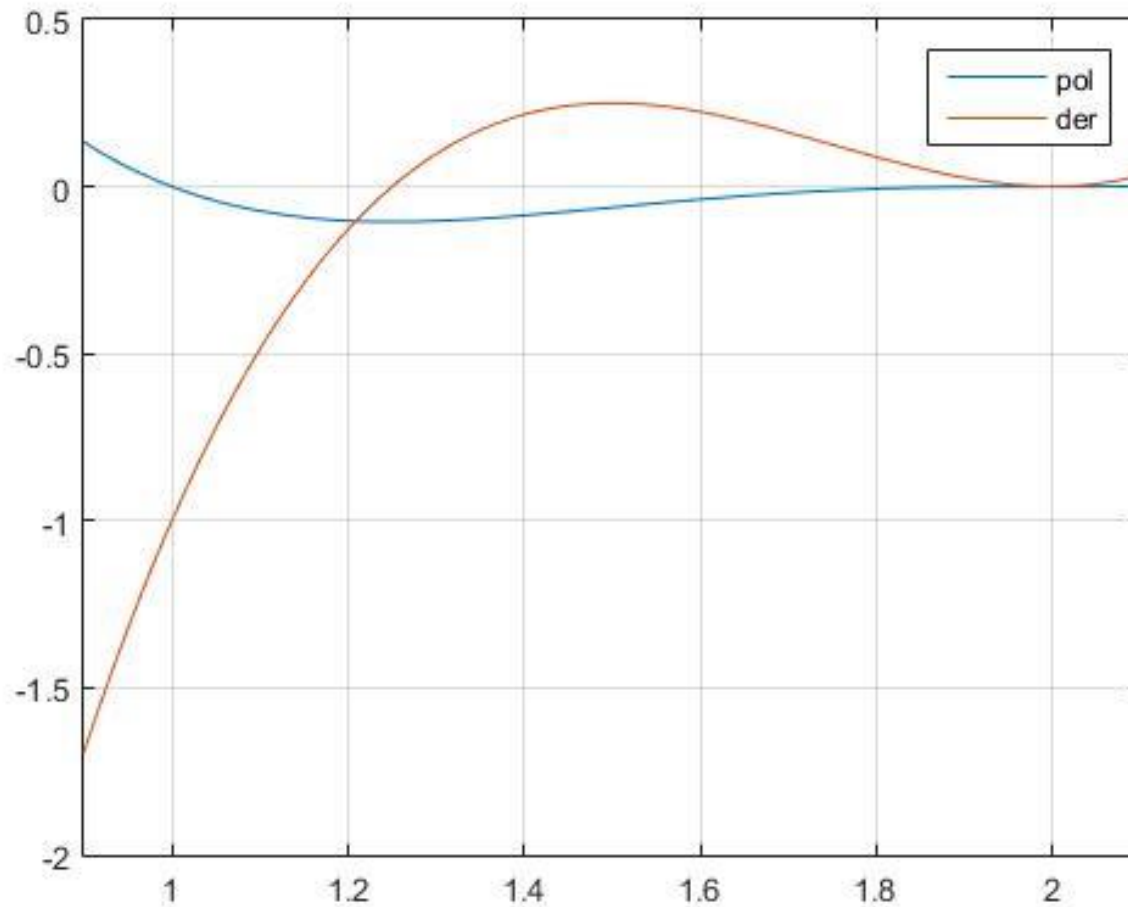
O zero igual a 1 é bem condicionado porque  $p'(1) = -1$ .



# O condicionamento das raízes

(3)

Quanto menor for  $|f'(r)|$  pior é o condicionamento da raiz  $r$  da equação  $f(x)=0$ .



# O condicionamento das raízes

(4)

No Matlab vamos calcular os coeficientes do polinómio mónico que tem o zero 2 com multiplicidade 9, isto é,  $p(x) = (x - 2)^9$ .

```
>> p=poly([2 2 2 2 2 2 2 2 2])
```

```
p =
```

```
    1   -18   144  -672   2016  -4032   5376  -4608   2304  -512
```

```
>> roots(p)
```

```
ans =
```

```
2.0689 + 0.0000i
```

```
2.0518 + 0.0449i
```

```
2.0518 - 0.0449i
```

```
2.0100 + 0.0668i
```

```
2.0100 - 0.0668i
```

```
1.9655 + 0.0566i
```

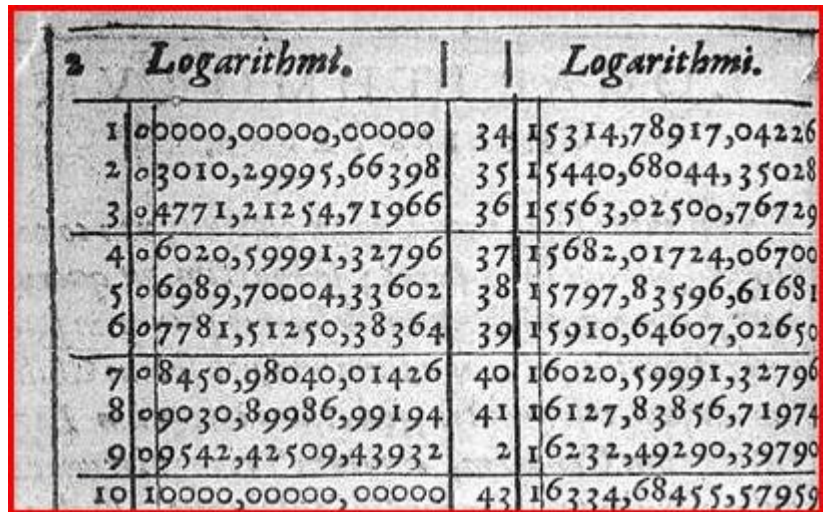
```
1.9655 - 0.0566i
```

```
1.9383 + 0.0218i
```

```
1.9383 - 0.0218i,
```

Se  $r$  é raiz múltipla da equação  $f(x) = 0$  então  $f'(r) = 0$  e a raiz é mal condicionada. O condicionamento é pior para multiplicidades maiores

### III. Interpolação polinomial de Lagrange

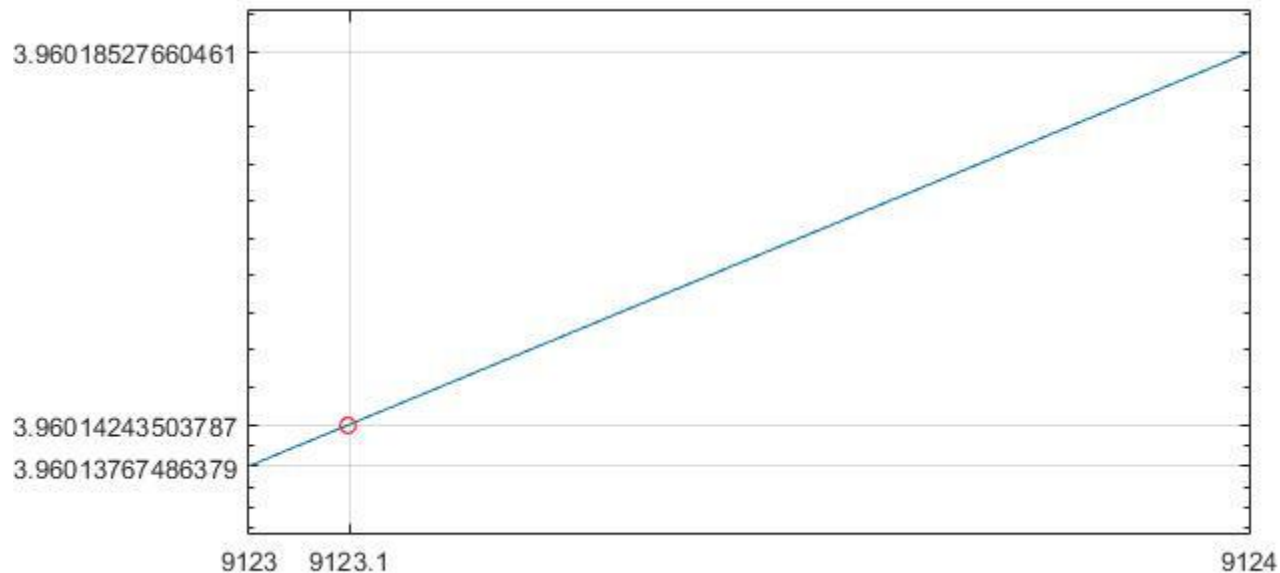


2		Logarithmi.			Logarithmi.
1	00000	,00000,00000	34	15314,78917,04226	
2	03010	,29995,66398	35	15440,68044,35028	
3	04771	,21254,71966	36	15563,02500,76729	
4	06020	,59991,32796	37	15682,01724,06700	
5	06989	,70004,33602	38	15797,83596,61681	
6	07781	,51250,38364	39	15910,64607,02650	
7	08450	,98040,01426	40	16020,59991,32796	
8	09030	,89986,99194	41	16127,83856,71974	
9	09542	,42509,43932	2	16232,49290,39790	
10	10000	,00000,00000	42	16334,68455,57959	

Em 1624, Henry Briggs publicou *Arithmetica Logarithmica* , uma tabela (tábua) com os logaritmos (com 14 algarismos decimais) dos números inteiros de 1 a 20 000 e de 90 001 até 100 000.

Usou, entre outros, o método da interpolação linear.

### III. Interpolação polinomial de Lagrange



```
>> z=interp1([9123,9124],log10([9123,9124]),9123.1)
```

dá a ordenada no ponto de abscissa 9123.1 situado na reta que passa pelos pontos (9123,log10(9123)) e (9124,log10(9124)).

# III. Interpolação polinomial de Lagrange

Existência e unicidade do polinómio interpolador de grau não superior a  $n$

Dados  $(n+1)$  pontos,  $(x_i, y_i), i = 0, 1, \dots, n, x_i$  distintos, existe e é único o polinómio  $\pi_n$  de grau menor ou igual a  $n$ , tal que

$$\pi_n(x_i) = y_i, \quad i = 0, 1, \dots, n. \quad (1)$$

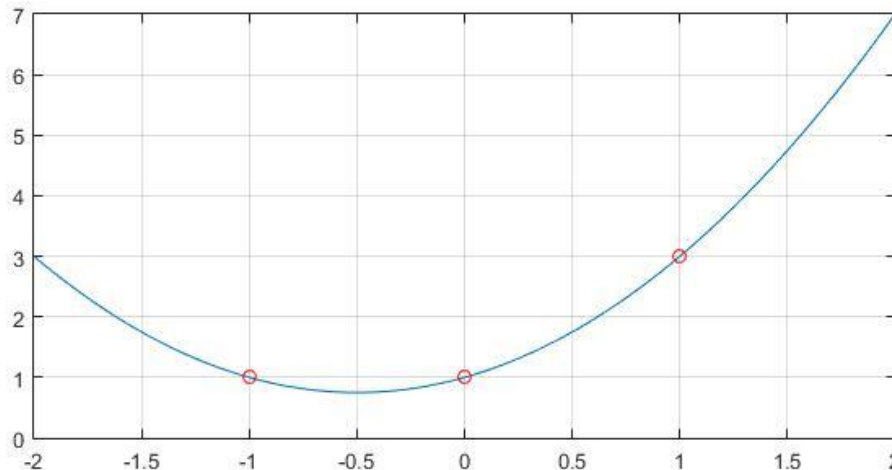
Se  $y_i = f(x_i)$ ,  $\pi_n$  é o polinómio interpolador de  $f$  nos **nós**  $x_i$ . Os  $y_i$  são os **valores nodais**.

$n=1$ : existe uma e uma só reta que passa por dois pontos

$n=2$ : existe uma e uma só parábola que passa por 3 pontos ...

Por exemplo, a parábola de equação  $y = x^2 + x + 1$  passa pelos pontos  $(-1,1)$ ,  $(0,1)$  e  $(1,3)$

Não existe outra parábola que passe por estes 3 pontos...



### III. Interpolação polinomial de Lagrange

## os coeficientes do polinómio interpolador

Com  $\pi_n(x) = a_1 x^n + a_2 x^{n-1} + \dots + a_n x + a_{n+1}$ ,  
as relações

$$\pi_n(x_i) = y_i, i = 0, 1, \dots, n.$$

Os coeficientes são a solução do sistema

[illegible]

ou, em notação matricial,

$$\begin{bmatrix} x_0^n & \cdots & 1 \\ \vdots & \ddots & \vdots \\ x_n^n & \cdots & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_{n+1} \end{bmatrix} = \begin{bmatrix} y_0 \\ \vdots \\ y_n \end{bmatrix}$$

# III. Interpolação polinomial de Lagrange

## A matriz de Vandermonde

O determinante da matriz  $\begin{bmatrix} x_0^n & \cdots & 1 \\ \vdots & \ddots & \vdots \\ x_n^n & \cdots & 1 \end{bmatrix}$  é diferente de zero se e só se os nós são distintos

(exercício da folha 4).

Neste caso, o sistema com  $n+1$  equações e  $n+1$  incógnitas é possível e determinado, isto é, tem uma e uma só solução  $a_1, a_2, \dots, a_{n+1}$ . Portanto, o polinómio de grau não superior a  $n$ , que satisfaz as  $n+1$  condições (1), existe e é único.

Exemplo: determinar o polinómio  $\pi_2$  de grau não superior a 2 que “passa” pelos 3 pontos (2,8), (3,11) e (4,14).

Os coeficientes de  $\pi_2(x) = a_1 x^2 + a_2 x + a_3$  são a solução de

$$\begin{bmatrix} 4 & 2 & 1 \\ 9 & 3 & 1 \\ 16 & 4 & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 8 \\ 11 \\ 14 \end{bmatrix}$$

```
>> a=vander([2,3,4])\[8;11;14]
```

```
a =  
0      Neste caso, o polinómio é  $\pi_2(x) = 3x + 2$ , de grau 1 (os pontos são colineares)  
3  
2
```

# III. Interpolação polinomial de Lagrange

## A fórmula interpoladora de Lagrange

---

Em geral, o que se pretende calcular é o valor do polinómio interpolador  $\pi_n$  em pontos dados. Para isto não é necessário determinar os coeficientes de  $\pi_n$ . Existem métodos mais eficientes (isto é, que requerem menos operações aritméticas).

Dados os nós  $x_0, x_1, \dots, x_n$  e os valores nodais  $y_0, y_1, \dots, y_n$ , escrevemos

$$\pi_n(x) = y_0 \cdot L_0(x) + y_1 \cdot L_1(x) + \dots + y_n \cdot L_n(x) = \sum_{i=0}^n y_i \cdot L_i(x)$$

onde

- $L_0(x_0)=1$  e  $L_0(x_j)=0$  para  $j \neq 0$
- $L_1(x_1)=1$  e  $L_1(x_j)=0$  para  $j \neq 1$
- ..
- $L_n(x_n)=1$  e  $L_n(x_j)=0$  para  $j \neq n$

$$L_0(x) = \frac{(x-x_1)(x-x_2)\dots(x-x_n)}{(x_0-x_1)(x_0-x_2)\dots(x_0-x_n)}, \dots, L_i(x) = \frac{(x-x_0)\dots(x-x_{i-1})(x-x_{i+1})\dots(x-x_n)}{(x_i-x_0)\dots(x_i-x_{i-1})(x_i-x_{i+1})\dots(x_i-x_n)} = \frac{\prod_{j=0, j \neq i}^n (x-x_j)}{\prod_{j=0, j \neq i}^n (x_i-x_j)}$$



# III. Interpolação polinomial de Lagrange

## A fórmula interpoladora de Lagrange (exemplo de aplicação)

---

Anteriormente, determinámos o polinómio  $\pi_2$  que “passa” nos pontos (2,8), (3,11) e (4,14). Sem usar a expressão encontrada, calculamos agora  $\pi_2(x)$  para  $x = 2.3$ .

$$L_0(x) = \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)}; \quad L_0(2.3) = \frac{(2.3-3)(2.3-4)}{(2-3)(2-4)} = 0.595$$

$$L_1(x) = \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)}; \quad L_1(2.3) = \frac{(2.3-2)(2.3-4)}{(3-2)(3-4)} = 0.510$$

$$L_2(x) = \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}; \quad L_2(2.3) = \frac{(2.3-2)(2.3-3)}{(4-2)(4-3)} = -0.1050$$

$$\pi_2(2.3) = 8*0.595 + 11*0.510 + 14*(-0.1050) = 8.9$$

Observe-se que os denominadores não dependem do ponto  $x$  em que se quer calcular  $\pi_2$  (só dependem dos nós) e portanto podem calcular-se uma única vez e usar-se para o cálculo de  $\pi_2$  em outros pontos  $x$ .

# III. Interpolação polinomial de Lagrange

A função poLagrange

$$\pi_n(x) = \sum_{i=0}^n y_i \cdot \frac{\prod_{j=0, j \neq i}^n (x - x_j)}{\prod_{j=0, j \neq i}^n (x_i - x_j)}$$

---

```
function px=poLagrange (xi,yi,x)
% Dados os nós e respectivos valores nodais na forma dos vectores
% xi e yi, respectivamente, usa a fórmula de Lagrange para calcular
% o valor do polinómio de grau não superior a n interpolador no ponto x
n=length(xi)-1;
px=0;
for i=1:n+1      % Li(x)=num/den;
    num=1; den=1;
    for j=1:n+1
        if (j~=i)
            num=num*(x-xi(j));
            den=den*(xi(i)-xi(j));
        end
    end
    px=px+yi(i)*(num/den);
end
```

# III. Interpolação polinomial de Lagrange

Interpolações da função  $\log_{10}$  com a função `poLagrange`

---

```
>> format long, xi=[9,10]; px=poLagrange(xi,log10(xi),9.1) % com dois nós
```

```
px = 0.958818258495392
```

```
>> xi=[9,9.5,10]; px=poLagrange(xi,log10(xi),9.1) % com 3 nós
```

```
px = 0.959035104700299
```

```
>> log10(9.1)
```

```
ans = 0.959041392321094
```

```
>> xi=[9123,9124]; px=poLagrange(xi, log10(xi), 9123.1) % com dois nós
```

```
px = 3.960142435037876
```

```
>> xi=[9123, 9123.5, 9124]; px=poLagrange(xi,log10(xi),9123.1) % com 3 nós
```

```
px = 3.960142435272663
```

```
>> log10(9123.1)
```

```
ans = 3.960142435272670
```

# III. Interpolação polinomial de Lagrange

A complexidade aritmética  $O(n^2)$  da fórmula de Lagrange

---

$$\pi_n(x) = \sum_{i=0}^n y_i \cdot \frac{\prod_{j=0, j \neq i}^n (x - x_j)}{\prod_{j=0, j \neq i}^n (x_i - x_j)}$$

Para cada  $i = 0, \dots, n$ , o cálculo do valor do numerador requer  $n$  subtrações e  $n-1$  multiplicações. O denominador outas tantas. O cálculo de cada  $L_i(x)$ ,  $i = 0, \dots, n$ , custa então  $4n - 1$  operações aritméticas.

Em rigor, o total de operações no cálculo de  $\pi_n(x)$  é de

$$(n + 1)((4n - 1) + 1) + n = 4n^2 + 5n$$

Esta não é a fórmula mais eficiente para calcular o valor de  $\pi_n(x)$

# III. Interpolação polinomial de Lagrange

## As diferenças divididas

---

Dados os nós  $x_0, x_1, \dots, x_n$  e os valores  $y_0 = f(x_0), y_1 = f(x_1), \dots, y_n = f(x_n)$ , de uma certa função  $f$ :

- Diferença dividida de primeira ordem relativa aos nós  $x_0$  e  $x_1$ :  $f[x_0, x_1] = \frac{f(x_0) - f(x_1)}{x_0 - x_1}$
- D. dividida de primeira ordem relativa aos nós  $x_1$  e  $x_2$ :  $f[x_1, x_2] = \frac{f(x_1) - f(x_2)}{x_1 - x_2}$
- ...
- D. D. de primeira ordem relativa aos nós  $x_{n-1}$  e  $x_n$ :  $f[x_{n-1}, x_n] = \frac{f(x_{n-1}) - f(x_n)}{x_{n-1} - x_n}$
  
- D.D. de 2ª ordem relativa aos nós  $x_0, x_1$  e  $x_2$ :  $f[x_0, x_1, x_2] = \frac{f[x_0, x_1] - f[x_1, x_2]}{x_0 - x_2}$
- D.D. de 2ª ordem relativa aos nós  $x_1, x_2$  e  $x_3$ :  $f[x_1, x_2, x_3] = \frac{f[x_1, x_2] - f[x_2, x_3]}{x_1 - x_3}$
- D.D. de 3ª ordem relativa aos nós  $x_0, x_1, x_2$  e  $x_3$ :  $f[x_0, x_1, x_2, x_3] = \frac{f[x_0, x_1, x_2] - f[x_1, x_2, x_3]}{x_0 - x_3}$

# III. Interpolação polinomial de Lagrange

A tabela das diferenças divididas (exemplo)

---

$$x_0=1, x_1=2, x_2=3, x_3=4, x_4=5 \text{ (nós)}$$

$$y_0=-5, y_1=-25, y_2=-47, y_3=-35, y_4=71$$

x	f(x)	f[,]	f[,]	f[,]	f[,]
1	-5				
2	-25	-20			
3	-47	-22	-1		
4	-35	12	17	6	
5	71	106	47	10	1

$$f(x_0)=-5$$

$$f[x_0, x_1]=-20$$

$$f[x_0, x_1, x_2]=-1$$

$$f[x_0, x_1, x_2, x_3]=6$$

$$f[x_0, x_1, x_2, x_3, x_4]=1$$

# III. Interpolação polinomial de Lagrange

## A fórmula interpoladora de Newton

---

De  $f[x, x_0] = \frac{f(x) - f(x_0)}{x - x_0}$ , com  $x \neq x_i$ , resulta

$$f(x) = f(x_0) + (x - x_0) f[x, x_0] \quad (1)$$

De  $f[x, x_0, x_1] = \frac{f[x, x_0] - f[x_0, x_1]}{x - x_1}$  resulta  $f[x, x_0] = f[x_0, x_1] + (x - x_1) f[x, x_0, x_1]$

e substituindo em (1) fica

$$f(x) = f(x_0) + (x - x_0) f[x_0, x_1] + (x - x_0)(x - x_1) f[x, x_0, x_1] \quad (2)$$

A expressão a bold define o polinómio interpolador de  $f$  de grau não superior a 1 nos nós  $x_0$  e  $x_1$  (porquê?)

De  $f[x, x_0, x_1, x_2] = \frac{f[x, x_0, x_1] - f[x_0, x_1, x_2]}{x - x_2}$  vem  $f[x, x_0, x_1] = f[x_0, x_1, x_2] + (x - x_2) f[x, x_0, x_1, x_2]$

e substituindo em (2) fica

$$f(x) = f(x_0) + (x - x_0) f[x_0, x_1] + (x - x_0)(x - x_1) f[x_0, x_1, x_2] + (x - x_0)(x - x_1)(x - x_2) f[x, x_0, x_1, x_2]$$

A expressão a bold define o polinómio interpolador de  $f$  de grau não superior a 2 nos nós  $x_0, x_1$  e  $x_2$ .

# III. Interpolação polinomial de Lagrange

## A fórmula interpoladora de Newton

---

Em geral tem-se

$$f(x) = \pi_n(x) + (x - x_0)(x - x_1) \dots (x - x_n) f[x, x_0, x_1, \dots, x_n]$$

onde

$$\pi_n(x) = f(x_0) + (x - x_0) f[x_0, x_1] + (x - x_0)(x - x_1) f[x_0, x_1, x_2] + \dots + (x - x_0)(x - x_1) \dots (x - x_{n-1}) f[x_0, x_1, \dots, x_n]$$

é o polinómio interpolador na forma de Newton, com diferenças divididas, e

$$(x - x_0)(x - x_1) \dots (x - x_n) f[x, x_0, x_1, \dots, x_n]$$

é o erro do polinómio interpolador



# III. Interpolação polinomial de Lagrange

## O erro do polinómio interpolador

---

O erro do polinómio interpolador pode expressar-se numa forma mais conveniente. Tem-se

$$f(x) = \pi_n(x) + (x - x_0)(x - x_1) \dots (x - x_n) \frac{f^{(n+1)}(\xi_x)}{(n+1)!}$$

onde  $\xi_x$  é um ponto (indeterminado) que está no intervalo  $I$  que contem os nós e o ponto  $x$ .

Majoração do erro: se  $|f^{(n+1)}(x)| \leq M$  para  $x \in I$ , então

$$|f(x) - \pi_n(x)| \leq (x - x_0)(x - x_1) \dots (x - x_n) \frac{M}{(n+1)!}$$

O polinómio  $W_n(x) = (x - x_0)(x - x_1) \dots (x - x_n)$ , cujos zeros são os nós de interpolação, é o polinómio nodal

# III. Interpolação polinomial de Lagrange

## Exemplos

---

### Exemplo 1

Com os nós  $x_0 = 1, x_1 = 1.2$  e os valores nodais  $y_0 = \log(1)$  e  $y_1 = \log(1.2)$  vamos usar interpolação linear para aproximar o valor de  $\log(1.1)$ . A fórmula de Lagrange dá

$$\pi_1(x) = \log(1) \times \frac{x-1.2}{1-1.2} + \log(1.2) \times \frac{x-1}{1.2-1}$$

```
>> x=1.1; log(1)*(x-1.2)/(1-1.2)+log(1.2)*(x-1)/(1.2-1)
ans = 0.0912
```

A fórmula de Newton dá  $\pi_1(x) = \log(1) + \frac{\log(1)-\log(1.2)}{1-1.2} (x-1)$

```
>> x=1.1; log(1)+(log(1)-log(1.2))/(1-1.2)*(x-1)
ans = 0.0912
```

# III. Interpolação polinomial de Lagrange

## Exemplos

---

### Exemplo 1(cont.)

Para o erro tem-se

$$\log(1.1) - \pi_1(1.1) = (1.1-1) \times (1.1-1.2) \times \frac{-1/\xi^2}{2} \text{ onde } \xi \text{ está entre 1 e 1.2}$$

Uma vez que  $|\frac{1}{\xi^2}| \leq \frac{1}{1^2}$ , resulta

$$|\log(1.1) - \pi_1(1.1)| \leq 0.1 \times 0.1 \times \frac{1}{2} = 0.005$$

Observe-se que  $\log(1.1)=0.0953\dots$ ,  $\pi_1(1.1)=0.0912$ ,

e 
$$|\log(1.1) - \pi_1(1.1)| = -0.0041 \dots$$

# III. Interpolação polinomial de Lagrange

## Exemplos

---

### Exemplo 2

Com os nós  $x = [1, 1.2, 1.3]$  e os valores nodais  $y = \log(x)$  vamos usar interpolação quadrática para aproximar o valor de  $\log(1.1)$ . A fórmula de Lagrange dá

$$\pi_2(x) = \log(1) \times \frac{(x-1.2)(x-1.3)}{(1-1.2)(1-1.3)} + \log(1.2) \times \frac{(x-1)(x-1.3)}{(1.2-1)(1.2-1.3)} + \log(1.3) \times \frac{(x-1)(x-1.2)}{(1.3-1)(1.3-1.2)}$$

```
>> poLagrange(x,log(x),1.1)
ans =0.0949
```

Para usar a formula de Newton, começamos por calcular a tabela das diferenças divididas

```
>> T=TabDifDiv(x,log(x))
```

T =

1	0	0	0
1.2	0.1823	0.9116	0
1.3	0.2624	0.8004	-0.3706

```
> x=1.1; 0+0.9116*(x-1)-0.3706*(x-1)*(x-1.2)
ans = 0.0949
```

# III. Interpolação polinomial de Lagrange

## Exemplos

---

Exemplo 2 (cont.)

$$\log(x) = \pi_2(x) + (x-1)(x-1.2)(x-1.3) \frac{2/\xi_x^3}{3!}$$

onde  $\xi_x$  está entre 1 e 1.3. Resulta que  $|2/\xi_x^3| \leq 2$  e

$$|\log(1.1) - \pi_2(1.1)| \leq (1.1-1)(1.1-1.2)(1.1-1.3) \frac{2}{3!}$$

```
>>x=1.1; (x-1)*(x-1.2)*(x-1.3)*2/6
```

```
ans = 6.6667e-04
```

Uma vez mais, podemos verificar que o erro é inferior a este majorante

```
>>log(1.1)-0.0949
```

```
ans = 4.1018e-04
```

# IV. Integração numérica

---

Teorema Fundamental do Cálculo Integral

$$\int_a^b f(x)dx = F(b) - F(a)$$

$F$  é uma primitiva de  $f$  em  $[a, b]$ , isto é,  $F'(x) = f(x)$  para todo  $x \in [a, b]$ .

Exemplo:

$$\int_0^1 e^{-x} dx = -e^{-1} - (-e^0) = 1 - e^{-1} \quad [F(x) = -e^{-x}]$$

$$\int_0^1 e^{-x^2} dx, \quad F=?$$

Situações em que se usam métodos numéricos para calcular integrais:

- Ainda que  $f$  seja definida por uma expressão analítica, não existe a primitiva  $F$  ou é difícil determiná-la;
- De  $f$  apenas se conhecem os valores que a função toma nalguns pontos

# IV. Integração numérica

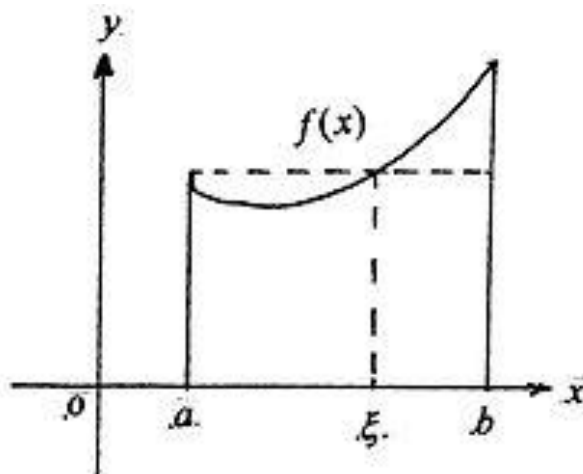
## Teorema do valor médio para integrais

Se  $f$  é contínua e  $g$  é integrável e não muda de sinal em  $[a, b]$ , então existe  $\xi \in [a, b]$  tal que

$$\int_a^b f(x)g(x)dx = f(\xi) \int_a^b g(x)dx$$

Para  $g(x)=1$

$$\int_a^b f(x)dx = f(\xi)(b - a)$$



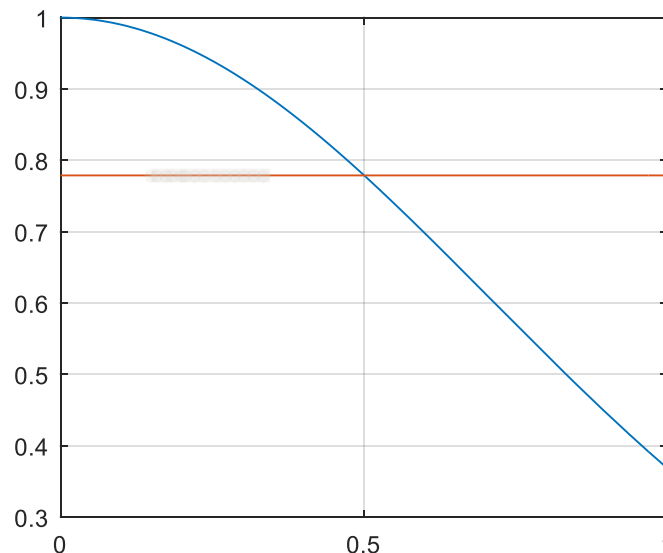
# IV. Integração numérica

## Regras de Newton-Cotes

Baseiam-se na interpolação polinomial

**n=0 (regra do ponto médio):** aproxima  $f$  pelo polinómio de grau 0 que a interpola no ponto médio  $\frac{a+b}{2}$ .

$$\int_a^b f(x)dx = \int_a^b f\left(\frac{a+b}{2}\right) dx + \text{erro de truncatura}$$
$$= (b-a)f\left(\frac{a+b}{2}\right) + \text{erro de truncatura}$$





# IV. Integração numérica

## Regras de Newton-Cotes

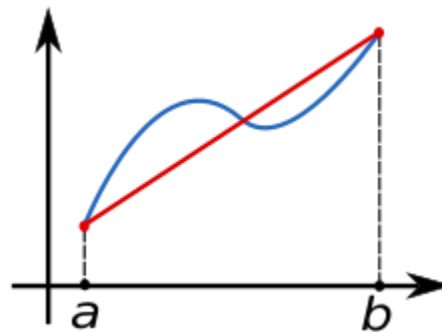
**n=1 (regra trapezoidal):** aproxima  $f$  pelo polinómio de grau não superior a 1 que a interpola nos extremos  $a$  e  $b$ . De

$$f(x) = f(a) \frac{x-b}{a-b} + f(b) \frac{x-a}{b-a} + (x-a)(x-b) \frac{f''(\xi_x)}{2}$$

resulta

$$\int_a^b f(x) dx = f(a) \int_a^b \frac{x-b}{a-b} dx + f(b) \int_a^b \frac{x-a}{b-a} dx + \frac{f''(\xi)}{2} \int_a^b (x-a)(x-b) dx$$

$$= h \frac{f(a)+f(b)}{2} - \frac{h^3}{12} f''(\xi) \quad \text{onde } h = b - a$$



A regra é de grau 1 (exata para polinómios de grau não superior a 1)

# IV. Integração numérica

## Regras de Newton-Cotes

**regra trapezoidal composta:** divide  $[a, b]$  em  $n$  sub-intervalos de igual amplitude  $h = \frac{b-a}{n}$  e aplica a regra em cada um dos subintervalos.

<https://upload.wikimedia.org/wikipedia/commons/7/7e/Trapezium2.gif>

Com  $x_i = a + i \cdot h$ ,  $i = 1, \dots, n-1$

$$\begin{aligned}\int_a^b f(x)dx &= \int_a^{x_1} f(x)dx + \int_{x_1}^{x_2} f(x)dx + \dots + \int_{x_{n-1}}^b f(x)dx \\ &= \frac{h}{2}[f(a) + f(x_1)] + \frac{h}{2}[f(x_1) + f(x_2)] + \dots + \frac{h}{2}[f(x_{n-1}) + f(b)] - \frac{h^3}{12} \sum_{i=0}^{n-1} f''(\eta_i) \\ &= \frac{h}{2}[f(a) + 2\sum_{i=1}^{n-1} f(x_i) + f(b)] - \frac{h^3}{12} n f''(\eta) \\ &= \frac{h}{2}[f(a) + 2\sum_{i=1}^{n-1} f(x_i) + f(b)] - \frac{h^2}{12} (b-a) f''(\eta)\end{aligned}$$

Se  $|f''(x)| \leq M$  para  $a \leq x \leq b$ , então o erro de truncatura tende para zero à medida que  $n$  cresce.

# IV. Integração numérica

## Regras de Newton-Cotes

**n=2 (regra de Simpson):** aproxima  $f$  pelo polinómio de grau não superior a 2 que a interpola nos extremos  $a$  e  $b$  e no ponto médio  $m = \frac{a+b}{2}$ .

De

$$f(x) = f(a) \frac{(x-m)(x-b)}{(a-m)(a-b)} + f(m) \frac{(x-a)(x-b)}{(m-a)(m-b)} + f(b) \frac{(x-a)(x-m)}{(b-a)(b-m)} + (x-a)(x-m)(x-b) \frac{f'''(\xi_x)}{3!}$$

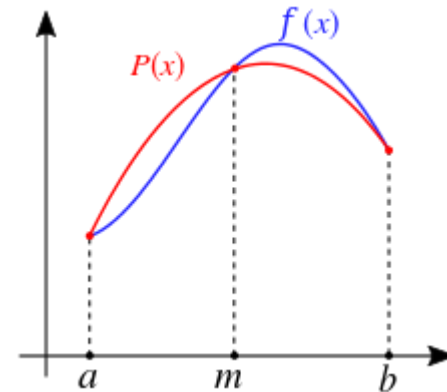
resulta

$$\int_a^b f(x) dx = f(a) \int_a^b \frac{(x-m)(x-b)}{(a-m)(a-b)} dx + f(m) \int_a^b \frac{(x-a)(x-b)}{(m-a)(m-b)} dx + f(b) \int_a^b \frac{(x-a)(x-m)}{(b-a)(b-m)} dx + \int_a^b \dots dx$$

$$= \frac{h}{3} [f(a) + 4f(m) + f(b)] + \frac{h^5}{90} f^{(iv)}(\eta)$$

$$\text{com } h = \frac{b-a}{2}.$$

A regra é de grau 3 (exata para polinómios de grau não superior a 3)



# IV. Integração numérica

## Regras de Newton-Cotes

**regra de Simpson composta** : divide  $[a, b]$  em  $n$  sub-intervalos ( $n$  par) de igual amplitude  $h = \frac{b-a}{n}$  e aplica a regra simples em cada um dos intervalos  $[a, x_2], [x_2, x_4], \dots, [x_{n-2}, b]$

[https://upload.wikimedia.org/wikipedia/commons/f/fc/Simpson%27s\\_One-Third\\_Rule.gif](https://upload.wikimedia.org/wikipedia/commons/f/fc/Simpson%27s_One-Third_Rule.gif)

$$\begin{aligned}\int_a^b f(x)dx &= \int_a^{x_2} f(x)dx + \int_{x_2}^{x_4} f(x)dx + \dots + \int_{x_{n-2}}^b f(x)dx \\ &= \frac{h}{3}[f(a) + 4f(x_1) + f(x_2)] + \frac{h}{3}[f(x_2) + 4f(x_3) + f(x_4)] + \dots + \frac{h}{3}[f(x_{n-2}) + 4f(x_{n-1}) + f(b)] + \\ &\quad \frac{h^5}{90} \sum_{i=1}^{n/2} f^{(iv)}(\eta_i) \quad \text{onde } a \leq \eta_1 \leq x_2, x_2 \leq \eta_2 \leq x_4, \dots, x_{n-2} \leq \eta_{n/2} \leq b\end{aligned}$$

$$\begin{aligned}&= \frac{h}{3}[f(a) + 4(f(x_1) + f(x_3) + \dots + f(x_{n-1})) + 2(f(x_2) + f(x_4) + \dots + f(x_{n-2})) + f(b)] + \\ &\quad \frac{h^4}{180}(b-a)f^{(iv)}(\eta)\end{aligned}$$

# IV. Integração numérica

## Regras de Newton-Cotes

---

Vamos calcular

$$I = \int_1^2 \log(x) dx$$

a) Regra simples dos trapézios

$$I \approx (2 - 1) \times \frac{\log(1) + \log(2)}{2} = 0.3466$$

Majoração do erro de truncatura

$$|I - 0.3466| \leq \frac{1}{12} \max_{1 \leq x \leq 2} \left| -\frac{1}{x^2} \right| = 1/12$$

b) Regra simples de Simpson

$$I \approx \frac{1/2}{3} [\log(1) + 4 \times \log(1.5) + \log(2)] = 0.3858$$

Majoração do erro de truncatura

$$|I - 0.3858| \leq \frac{(1/2)^5}{90} \max_{1 \leq x \leq 2} \left| -\frac{6}{x^4} \right| = 0.0021$$

# IV. Integração numérica

A função trapezios.m

---

```
function T=trapezios(f, a, b, n )
```

```
% Usa a regra composta dos trapézios para aproximar o valor do integral de f
```

```
% no intervalo [a,b]; divide o intervalo em n subintervalos de igual amplitude;
```

```
h=(b-a)/n;
```

```
T=(f(a)+f(b))/2;
```

```
x=a+h:h:b-h;
```

```
T=T+sum(f(x));
```

```
T=T*h;
```

# IV. Integração numérica

A função `simpson.m`

---

```
function Q=simpson(f, a, b, n)
% Usa a regra composta de Simpson para aproximar o valor do integral de f
% no intervalo [a,b];
% divide o intervalo em n sub-intervalos de igual amplitude; n deve ser par

h=(b-a)/n;
Q=f(a)+f(b);
x=a+h:2*h:b-h;
Q=Q+4*sum(f(x));
x=a+2*h:2*h:b-2*h;
Q=Q+2*sum(f(x));
Q=Q*h/3;
```

## V. Sistemas de equações lineares

Com  $m$  equações e  $n$  incógnitas

[illegible]

Em notação matricial,

$$\begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} \textcolor{red}{x}_1 \\ \vdots \\ \textcolor{red}{x}_n \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}$$

$$Ax = b$$



# V. Sistemas de equações lineares

---

Um sistema pode ter uma e uma só solução  $x$ , nenhuma solução, um número infinito de soluções.

A solução do sistema seguinte são as coordenadas no espaço tridimensional dos pontos que estão na interseção dos 3 planos cujas equações são as dadas

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = b_3 \end{cases}$$

A interseção dos planos pode ser

- ❑ Um ponto (sistema possível e determinado)
- ❑ Uma reta (sistema possível e indeterminado)
- ❑ Um plano (se os 3 planos coincidem)
- ❑ Vazia se pelo menos dois dos planos são paralelos ou um deles é paralelo à reta de interseção dos outros dois

## V. Sistemas de equações lineares

---

No exemplo seguinte as duas primeiras equações representam planos paralelos

$$\begin{cases} 2x_1 - 3x_2 + 7x_3 = 1 \\ 2x_1 - 3x_2 + 7x_3 = 0 \\ 3x_1 + 6x_2 + 5x_3 = 0 \end{cases}$$

e o sistema é impossível (isto é, não há nenhum ponto que pertença a todos os planos).

Subtraindo a primeira equação da segunda (esta é uma OPERAÇÃO de EQUIVALÊNCIA) resulta  $0 = -1$ .

## V. Sistemas de equações lineares

---

$$\text{O sistema } \begin{cases} x_1 + \frac{5}{3}x_2 + x_3 = 1 \\ 3x_1 + 6x_2 - 5x_3 = 0 \\ 4x_1 + \frac{23}{3}x_2 - 4x_3 = 1 \end{cases}$$

é indeterminado. Adicionando à segunda equação a primeira multiplicada por -3 e adicionando à terceira equação também a primeira multiplicada por -4, resulta o sistema equivalente

$$\begin{cases} x_1 + \frac{5}{3}x_2 + x_3 = 1 \\ x_2 - 8x_3 = -3 \\ x_2 - 8x_3 = -3 \end{cases}$$

Podemos fixar  $x_3 = \alpha$ , qualquer. Resulta  $x_2 = -3 + 8\alpha$  e  $x_1 = 1 - \frac{5}{3}x_2 - x_3 = 1 - \frac{5}{3}(-3 + 8\alpha) - \alpha$

$$\text{A equação da reta é } \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 6 \\ -3 \\ 0 \end{pmatrix} + \alpha \begin{pmatrix} -43/3 \\ 8 \\ 1 \end{pmatrix}$$

# V. Sistemas de equações lineares

---

Com  $n$  equações e  $n$  incógnitas (a matriz  $A$  é quadrada)

$$\begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}$$

a solução existe e é única se e só se  $A^{-1}$  existe.

De  $Ax = b$  resulta  $A^{-1}Ax = A^{-1}b$  e

$$x = A^{-1}b$$

Nota importante: esta relação só tem interesse teórico; na prática nunca se calcula a inversa  $A^{-1}$  para resolver o sistema  $Ax = b$

# V. Sistemas de equações lineares

as matrizes de eliminação Gaussiana

Dado o sistema

$$Ax = b,$$

com  $n$  equações e  $n$  incógnitas, se  $M$  é uma matriz de ordem  $n$  tal que  $M^{-1}$  existe, então os sistemas  $Ax = b$  e  $MAx = Mb$  têm a mesma solução (são equivalentes).

As operações de equivalência efetuam-se sobre as linhas da matriz ampliada (cada linha representa uma equação do sistema). Para  $n = 4$  e assumindo que  $a_{11} \neq 0$ ,

$$\text{com } M_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -\frac{a_{21}}{a_{11}} & 1 & 0 & 0 \\ -\frac{a_{31}}{a_{11}} & 0 & 1 & 0 \\ -\frac{a_{41}}{a_{11}} & 0 & 0 & 1 \end{bmatrix} \quad \text{resulta} \quad M_1 \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & | & b_1 \\ a_{21} & a_{22} & a_{23} & a_{24} & | & b_2 \\ a_{31} & a_{32} & a_{33} & a_{34} & | & b_3 \\ a_{41} & a_{42} & a_{43} & a_{44} & | & b_4 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & | & b_1 \\ 0 & a_{22} & a_{23} & a_{24} & | & b_2 \\ 0 & a_{32} & a_{33} & a_{34} & | & b_3 \\ 0 & a_{42} & a_{43} & a_{44} & | & b_4 \end{bmatrix}$$

Este é o primeiro passo de redução ( $k = 1$ ) no processo de reduzir a matriz do sistema à forma triangular superior.

*Para  $i=2,3,4$ : (i.e,  $i=k+1, \dots, n$ )*

$$-\frac{a_{i1}}{a_{11}} [a_{11} \ a_{12} \ a_{13} \ a_{14} \ | \ b_1] + [a_{i1} \ a_{i2} \ a_{i3} \ a_{i4} \ | \ b_i] \rightarrow [0 \ a_{i2} \ a_{i3} \ a_{i4} \ | \ b_i]$$

# V. Sistemas de equações lineares

as matrizes de eliminação Gaussiana

Assumindo que  $a_{22} \neq 0$

$$\text{com } M_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & \frac{-a_{32}}{a_{22}} & 1 & 0 \\ 0 & \frac{-a_{42}}{a_{22}} & 0 & 1 \end{bmatrix} \quad \text{resulta} \quad M_2 \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & | & b_1 \\ 0 & a_{22} & a_{23} & a_{24} & | & b_2 \\ 0 & a_{32} & a_{33} & a_{34} & | & b_3 \\ 0 & a_{42} & a_{43} & a_{44} & | & b_4 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & | & b_1 \\ 0 & a_{22} & a_{23} & a_{24} & | & b_2 \\ 0 & 0 & a_{33} & a_{34} & | & b_3 \\ 0 & 0 & a_{43} & a_{44} & | & b_4 \end{bmatrix}$$

Este é o segundo passo de redução ( $k = 2$ ) no processo de reduzir a matriz do sistema à forma triangular superior.

*Para  $i=3,4$ : (i.e,  $i=k+1, \dots, n$ )*

$$-\frac{a_{i2}}{a_{22}} [0 \ a_{22} \ a_{23} \ a_{24} \ | \ b_2] + [0 \ a_{i2} \ a_{i3} \ a_{i4} \ | \ b_i] \rightarrow [0 \ 0 \ a_{i3} \ a_{i4} \ | \ b_i]$$

# V. Sistemas de equações lineares

as matrizes de eliminação Gaussiana

Finalmente, assumindo que  $a_{33} \neq 0$  (são  $n - 1$  passos de redução)

$$\text{com } M_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{-a_{43}}{a_{33}} & 1 \end{bmatrix} \quad \text{resulta} \quad M_3 \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & | & b_1 \\ 0 & a_{22} & a_{23} & a_{24} & | & b_2 \\ 0 & 0 & a_{33} & a_{34} & | & b_3 \\ 0 & 0 & a_{43} & a_{44} & | & b_4 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & | & b_1 \\ 0 & a_{22} & a_{23} & a_{24} & | & b_2 \\ 0 & 0 & a_{33} & a_{34} & | & b_3 \\ 0 & 0 & 0 & a_{44} & | & b_4 \end{bmatrix}$$

Este é o terceiro e último passo de redução ( $k = 3$ ) (são  $n - 1$  passos de redução)

*Para  $i=4$ : (i.e,  $i=k+1, \dots, n$ )*

$$-\frac{a_{i3}}{a_{33}} [0 \ 0 \ a_{33} \ a_{34} \ | \ b_3] + [0 \ 0 \ a_{i3} \ a_{i4} \ | \ b_i] \rightarrow [0 \ 0 \ 0 \ a_{i4} \ | \ b_i]$$

# V. Sistemas de equações lineares

o método de eliminação de Gauss (implementação Matlab)

---

```
function x=GaussElim (A, b)
% resolve o sistema Ax=b pelo método de eliminação de Gauss sem pivotação
% usa a function STriangular que implementa o método de substituição
% inversa para matrizes triangulares superiores;
n=length(b);
for k=1:n-1
    for i=k+1:n
        m=A(i,k)/A(k,k);
        A(i,k:n)=A(i,k:n)-m*A(k,k:n);
        b(i)=b(i)-m*b(k);
    end
end
x=STriangular(A,b);
```



# V. Sistemas de equações lineares

o método de eliminação de Gauss (implementação Matlab)

---

```
function x=GaussElim (A, b)
% resolve o sistema Ax=b pelo método de eliminação de Gauss sem pivotação
% usa a function STriangular que implementa o método de substituição
% inversa para matrizes triangulares superiores;
n=length(b);
for k=1:n-1
    for i=k+1:n
        m=A(i,k)/A(k,k);
        A(i,k:n)=A(i,k:n)-m*A(k,k:n);
        b(i)=b(i)-m*b(k);
    end
end
x=STriangular(A,b);
```

# V. Sistemas de equações lineares

o método de substituição inversa (implementação Matlab)

---

```
function x=STriangular(A,b)
% implementa o método de substituição inversa para resolver o sistema
% Ax=b, sendo A uma matriz triangular superior

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

# V. Sistemas de equações lineares

## condicionamento de um sistema

---

O sistema  $Ax = b$  é mal condicionado se perturbações nos dados (isto é, nas entradas de  $A$  e/ou de  $b$ ) produzirem erros maiores na solução  $x$ .

O número de condição é  $\kappa(A) = \|A\| \cdot \|A^{-1}\|$ . O erro relativo na solução  $x$  pode chegar a ser tão grande quanto o produto de  $\kappa(A)$  pelos erros relativos em  $A$  e  $b$ .

Exemplos no Matlab usando as matrizes de Hilbert

1	1/2	1/3	1/4	1/5	...
1/2	1/3	1/4	1/5	1/6	...
1/3	1/4	1/5	1/6	1/7	...
1/4	1/5	1/6	1/7	1/8	...
1/5	1/6	1/7	1/8	1/9	...
...	...	...	...	...	...

# V. Sistemas de equações lineares

## condicionamento de um sistema (exemplo 1)

---

```
>> A=hilb(5); b=A*ones(5,1)
```

A solução exata do sistema  $Ax = b$  é

```
>> x=ones(5,1)
```

Formamos a matriz  $\tilde{A}$  introduzindo erros na matriz original

```
>> Atil=A+1e-5*rand(5)
```

Atil =

1.0000	0.5000	0.3333	0.2500	0.2000
0.5000	0.3333	0.2500	0.2000	0.1667
0.3333	0.2500	0.2000	0.1667	0.1429
0.2500	0.2000	0.1667	0.1429	0.1250
0.2000	0.1667	0.1429	0.1250	0.1111

(em format short não se visualiza qualquer diferença relativamente à matriz inicial)

# V. Sistemas de equações lineares

condicionamento de um sistema (continuação do exemplo 1)

---

O erro relativo na matriz é dado por

```
>> norm(A-Atil)/norm(A)
```

```
ans = 1.4628e-05
```

A solução do sistema perturbado é

```
>> xtil=Atil\b
```

```
xtil = 0.9990, 1.0213, 0.8946, 1.1776, 0.9054    (transposto)
```

O erro relativo na solução é

```
>> norm(x-xtil)/norm(x)
```

```
ans = 0.1020
```

O número de condição é

```
>> norm(A)*norm(inv(A))
```

```
ans = 4.7661e+05    (nota: 4.7661e+05*1.4628e-05=6.9719, os erros em xtil podiam  
ser maiores ainda)
```

## V. Sistemas de equações lineares

### condicionamento de um sistema (exemplo 2)

---

```
>> A=hilb(10); b=A*ones(10,1);
```

A solução exata do sistema  $Ax = b$  é

```
>> x=ones(10,1) mas
```

```
>> x=A\b
```

x =

1.0000

O que origina estes erros uma vez que, ao contrário do exemplo anterior, não introduzimos perturbações na matriz A?

1.0000

1.0000

1.0000

Uma vez que nem todas as entradas da matriz têm representação exata, existem erros de arredondamento (recorde-se que, em valor relativo, são majorados por  $\text{eps}/2$ ).

0.9999

1.0003

0.9995

Estes erros são ampliados por  $\kappa(A) = \|A\| \cdot \|A^{-1}\| = 1.6024e+13$

1.0004

0.9998

1.0001

(nota:  $1.6024e+13 * \text{eps}/2 = 0.0018$ )

# V. Sistemas de equações lineares

Instabilidade numérica da eliminação de Gauss (exemplo)

---

O sistema  $\begin{cases} \varepsilon x_1 + x_2 = 10 \\ x_1 + x_2 = 1 \end{cases}$  tem a solução  $\begin{cases} x_1 = \frac{9}{\varepsilon-1} \\ x_2 = \frac{\varepsilon-10}{\varepsilon-1} \end{cases}$  o que para  $\varepsilon \approx 0$  dá  $\begin{cases} x_1 \approx -9 \\ x_2 \approx 10 \end{cases}$

No Matlab, com  $\varepsilon = 2^{-52}$ ,

```
>> format long, A=[eps 1; 1 1]; b=[10; 1]; A\b
```

ans =

```
-9.0000000000000002  
10.0000000000000002
```

Mas

```
>> x=GaussElim(A,b)
```

ou seja, o valor de  $x_1$  está errado. Porquê?

x =

```
-8.0000000000000000  
10.0000000000000002
```

# V. Sistemas de equações lineares

Instabilidade numérica da eliminação de Gauss (exemplo)

---

O sistema  $\begin{cases} \varepsilon x_1 + x_2 = 10 \\ x_1 + x_2 = 1 \end{cases}$  tem a solução  $\begin{cases} x_1 = \frac{9}{\varepsilon-1} \\ x_2 = \frac{\varepsilon-10}{\varepsilon-1} \end{cases}$  o que para  $\varepsilon \approx 0$  dá  $\begin{cases} x_1 \approx -9 \\ x_2 \approx 10 \end{cases}$

No Matlab, com  $\varepsilon = 2^{-52}$ ,

```
>> format long, A=[eps 1; 1 1]; b=[10; 1]; A\b
```

ans =

```
-9.0000000000000002  
10.0000000000000002
```

Mas

```
>> x=GaussElim(A,b)
```

ou seja, o valor de  $x_1$  está errado. Porquê?

x =

```
-8.0000000000000000  
10.0000000000000002
```



# V. Sistemas de equações lineares

## Instabilidade numérica da eliminação de Gauss (exemplo)

Para eliminar a entrada **1**, em  $\left[ \begin{array}{cc|c} 2^{-52} & 1 & 10 \\ \mathbf{1} & 1 & 1 \end{array} \right]$  o multiplicador é  $-1/2^{-52} = -2^{52}$

A ocorrência de um multiplicador tão grande vai aumentar muito os erros de arredondamento nas operações subsequentes. O problema fica resolvido se trocarmos as equações.

Agora, para eliminar a entrada  **$2^{-52}$** , em  $\left[ \begin{array}{cc|c} 1 & 1 & 1 \\ \mathbf{2^{-52}} & 1 & 10 \end{array} \right]$  o multiplicador é  $-2^{-52}$ , muito pequeno.

```
>> A=[1 1; eps 1]; b=[1; 10]; x=GaussElim(A,b)
```

```
ans =
```

```
-9.0000000000000002
```

```
10.0000000000000002
```

que é a solução dada por  $A \backslash b$  e está correta.

# V. Sistemas de equações lineares

## A pivotação parcial

No início do  $k$ -ésimo passo de redução, determina-se o pivot que é a maior, em valor absoluto, das entradas na  $k$ -ésima coluna abaixo da diagonal principal. Designando por  $p$  a linha onde se encontra o pivot (note-se que é  $p \geq k$ , trocam-se as linhas  $p$  e  $k$  da matriz ampliada. Procede-se à eliminação usual.

Exemplo

1º passo de redução

$$\begin{array}{ccc|c} 1.0000 & 1.0000 & 3.0000 & -1 \\ 2.0000 & 2.0000 & 2.0000 & 0 \\ 0.1000 & 5.0000 & 3.0000 & 2 \end{array} \rightarrow \begin{array}{ccc|c} 1.0000 & 1.0000 & 3.0000 & -1 \\ 0.1000 & 5.0000 & 3.0000 & 2 \\ 2.0000 & 2.0000 & 2.0000 & 0 \end{array} \rightarrow \begin{array}{ccc|c} 1.0000 & 1.0000 & 3.0000 & -1 \\ 0 & 4.9000 & 2.9000 & 2 \\ 0 & 0 & 2.0000 & -1 \end{array}$$

2º passo de redução

$$\begin{array}{ccc|c} 2.0000 & 2.0000 & 2.0000 & 0 \\ 0 & 4.9000 & 2.9000 & 2 \\ 0 & 0 & 2.0000 & -1 \end{array}$$

A substituição inversa dá:

$$\begin{aligned} x(3) &= b(3)/A(3,3) = -0.5 \\ x(2) &= (b(2) - A(2,3)*x(3))/A(2,2) = 0.7041 \\ x(1) &= (b(1) - A(1,2)*x(2) - A(1,3)*x(3))/A(1,1) = -0.2041 \end{aligned}$$

# V. Sistemas de equações lineares

A função GaussElimPP

---

```
function x=GaussElimPP(A,b)
% resolve o sistema Ax=b pelo método de eliminação de Gauss
% com pivotação parcial(PP);
n=length(b);
for k=1:n-1
    [~,p]=max(abs(A(k:n,k))); p=p+k-1; % indice corrigido da linha pivotal
    if p~=k
        A([k p],k:n)=A([p k],k:n) % troca linhas k <--> p da matriz A
        b([k p])=b([p k]); % troca b(k) <--> b(p)
    end
    for i=k+1:n
        m(i,k)=A(i,k)/A(k,k);
        for j=k+1:n
            A(i,j)=A(i,j)-m(i,k)*A(k,j);
        end
        b(i)=b(i)-m(i,k)*b(k);
    end
end
end
x=STriangular(A,b);
```

# V. Sistemas de equações lineares

## GaussElimPP vs GaussElim (exemplo)

---

Em muitos casos a pivotação parcial não seria necessária

```
>> format long, A=rand(3); b=A*ones(3,1);  
>> x=GaussElim(A,b); xPP=GaussElimPP(A,b); [x xPP]  
ans =
```

```
1.0000000000000000  1.0000000000000000  
1.0000000000000000  1.0000000000000000  
1.0000000000000000  1.0000000000000000
```

Mas é necessária nos casos em que ocorrem multiplicadores de grande valor absoluto.

De uma maneira geral deve usar-se pivotação parcial. É o que acontece na função implementada no Matlab.