Erros de arredondamento

- O conjunto $\mathbb{R}$ dos números reais é infinito, contínuo e ilimitado. O subconjunto $\mathscr{F}$ 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 $\mathscr{F}$; 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 confito que resultou da invasão do Koweit por tropas iraquianas (Guerra do Golfo), uma bateria de misseis norte-americanos Patriot cuja missão era a de proteger uma instalação militar em Daharan, na Arábia Saudita, falhou a intercepção 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 misseis 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:

$$\mathbf{fl(0.1)} = \mathbf{0.0001100110011001100}\,1100\ldots$$

Erro de arredondamento (absoluto)

$$|\mathbf{0.1} - \boldsymbol{fl}(\mathbf{0.1})| = \mathbf{2^{-24} + 2^{-25} + 2^{-28}} \ldots \backsimeq \mathbf{9.5} * \mathbf{10^{-8}}$$

Continuação do exemplo

- É este pequeno erro $\mathbf{9.5 * 10^{-8}}$ 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 * $\mathbf{9.5 * 10^{-8}}$ segundos $\backsimeq$ 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 mas, 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 à unidade; em geral, >> ones(m; n) produz uma matriz com m linhas e n colunas e entradas todas iguais à unidade].

>> sum(x)

produz o resultado **1.0000**

>> ans-1

dá **-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)

Cálculo numérico *versus* cálculo algébrico

- *Um sistema algébrico 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 algébrico, 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 algébrico 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+algébrico].

O ponto flutuante

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

$$x = \pm \mathrm{m} * \beta^{e} \qquad (1)$$

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

- Valores típicos de β 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,

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

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 aspeto (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

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

- Exemplo

$$(0.1)_{10}=(0.0001100110011001100110011\ 1100\dots\ )_{2}$$

tem a representação normalizada (com 32 bits)

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

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

A norma IEEE754 (sobre o **eps**ilon)

No formato duplo, o sucessor do número 1

$$1=(\mathbf{1}.0\ldots00)\ *\ 2^{0}$$

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

$$(\mathbf{1}.0\ldots01)\ *\ 2^{0}=1+\ 2^{-52}$$

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 =

     1
```

Percorrendo $\mathscr{F}$

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

$$2=(\mathbf{1}.0\ldots00)\ *\ 2^1$$

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

é $$(\mathbf{1}.0\ldots01)\ *\ 2^1=(1+\text{eps})*2=2+2\text{eps}$$

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

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

é $$|x_+-x_-|=2^{e-52}$$

Majoração do erro absoluto de arredondamento

- Se $x_- < x < x_+$

resulta da expressão anterior que

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

- 2^{e-52} é o majorante do erro absoluto devido ao arredondamento
- Para números grandes este erro absoluto pode ser proporcionalmente grande;

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

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

o que mostra que o erro absoluto pode ser grande (e ainda maior do que isto para expoentes maiores).

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

Majoração do erro relativo de arredondamento para números normais

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

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

- Tem-se

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

isto é, o erro relativo é majorado por eps, desde que x não seja, em valor absoluto, menor do que 2^{-1022}(Porquê?)

- 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

$$\mathbf{2^8} = 256$$

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 $\mathbf{-126 \leq\ e\ (inteiro)\ \leq 127}$

- Formato duplo $\mathbf{-1022 \leq\ e\ (\boldsymbol{inteiro})\ \leq 1023}$

O maior número representável na norma IEEE 754

$$(\mathbf{1.1111111111111111111111111111111111111111111111111111})* \mathbf{2^{1023}}$$
$$= \mathbf{2^{1023}} + \mathbf{2^{1022}} + \ldots + \mathbf{2^{971}}$$

```
>> 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

- $m = 1 + 2^{-1} + 2^{-2} + \ldots + 2^{-51} + 2^{-52}$

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

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

- realmax=$(2 - 2^{-52}) * 2^{1023} = 2^{1024} - 2^{971}$

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

     0

>> 2^1024
ans =

   Inf
```

O realmin

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

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

ans =

     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

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

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

...

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

>> 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 =
   1
-------------------------
```

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

```
ans =
  0
```

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

```
ans =

    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')

‘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
% arredonamento previsto 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

Algarismos significativos

Nas representações seguintes o número π 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)

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

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

Algarismos significativos corretos

Se

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

(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 $|\boldsymbol{x}| \geq d_{-1}{*}10^{e}$,

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

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

Exemplo no Matlab

$$\boldsymbol{\pi = 3.1415926535\ldots}$$

Com **t** algarismos significativos corretos:

- t=3, $\bar{\pi} = \mathbf{3.14}$

> (pi-3.14)/pi dá 5.0696e-04 (< 5*10^-3)

- t=4, $\bar{\pi} = \mathbf{3.142}$

>> (pi-3.141)/pi dá 1.8865e-04 (<5*10^-4)

- t=5, $\bar{\pi} = \mathbf{3.1416}$

>> (pi-3.1416)/pi dá -2.3384e-06 (<5*10^-5)

O cancelamento subtrativo (1)

Perda de algarismos significativos na subtração de números

Exemplo

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

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

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

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

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

e

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

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 ($sin(x)^2+cos(x)^2=1$)

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

ans =

    1.000000082740371e-10


ans =

    9.999999999666671e-11
```

Questão: qual destes números tem mais algarismos significativos corretos?

O cancelamento subtrativo (3)

Resposta: é no resultado dado por sin(x)^2

Ocorre cancelamento subtrativo na diferença 1-cos(x)^2.

>> cos(1e-5)^2

ans =

0.999999999900000

>> 1-cos(1e-5)^2

ans =

1.000000082740371e-10

O erro relativo no cancelamento subtrativo (1)

1- cos(1e-5)^2 dá **1.000000082740371e-10**

e **sin(1e-5)^2 dá 9.999999999666671e-11**

Tomando sin(1e-5)^2 como valor exato, o erro absoluto em 1- cos(1e-5)^2 é pequeno

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

ans = 3.247895369989818e-16

mas o erro relativo é muito maior

> ans/sin(x)^2

ans =

3.247895370098080e-06

O erro relativo no cancelamento subtrativo (2)

1- cos(1e-5)^2 dá 1.000000082740371e-10
sin(1e-5)^2 **dá** 9.999999999666671e-11

Tomando sin(1e-5)^2 como valor exato, o erro absoluto em 1- cos(1e-5)^2 é pequeno

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

ans = 8.277370427339253e-18

mas o erro relativo é muito maior

> ans/sin(x)^2

ans =

8.277370427615165e-08

Propagação do erro relativo

Erros relativos grandes podem gerar erros absolutos também grandes

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

ans =

   1.000000082773704
```

O resultado exato é 1, o erro absoluto é

```
>> abs(1-ans)

 ans =8.277370433518172e-08
```

Condicionamento 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

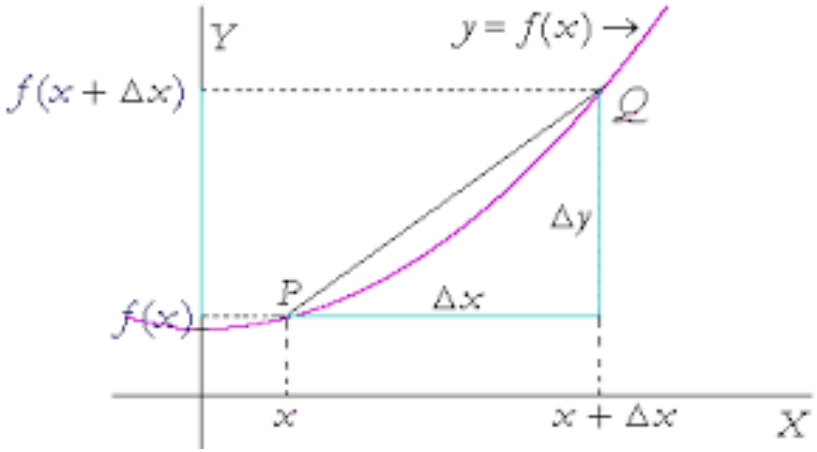
```
>> x=10; deltaX=1e-5; exp(x+deltaX)-exp(x)

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



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

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

$$f(x + \Delta x) = f(x) + f'(x).\ \Delta x + \frac{f''(x)}{2}(\Delta x)^2 + \ldots + \frac{f^{(k)}(x)}{k!}(\Delta x)^k + \cdots$$

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

O número de condição relativo de uma função num ponto

De

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

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

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

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

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

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

$\Delta y \approx |\boldsymbol{f'(x)}|. \Delta x = \boldsymbol{e^{10}} * 10^{-5} = 0.2203 \ldots$ (erro absoluto cresce $e^{10} \approx 22026$ vezes)

$\left|\frac{\Delta y}{y}\right| \approx |\boldsymbol{x}|. \left|\frac{\Delta x}{x}\right| = \mathbf{10} * 10^{-6}$ (erro relativo cresce 10 vezes)

Números de condição relativos e perda de algarimos 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 =

1.000010000000000 x+deltaX aproxima x com 5 algarimos corretos

ans =

2.718281828459046 número de condição é |x|=1

ans =

2.718309011413245 exp(x+deltaX) aproxima exp(x) com 5 algarimos corretos

Exemplo 2

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

ans =

1.000010000000000e+02 x+deltaX aproxima x com 5 algarimos corretos

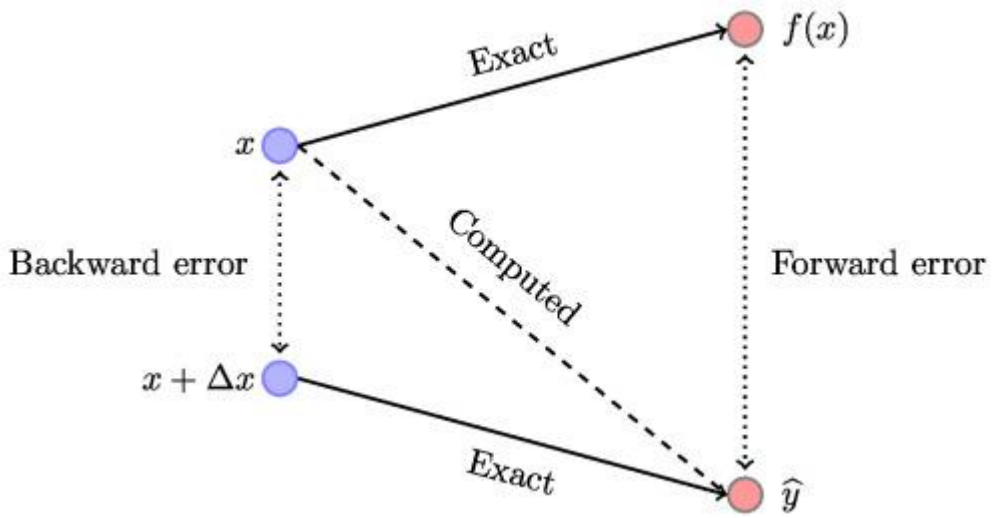
ans =

2.688117141816136e+43 número de condição é |x|=100

ans =

2.690806603464667e+43 exp(x+deltaX) aproxima exp(x) com 3 algarimos 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 =
   6.666666666671666e+05
```

Séries de Taylor (1)

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

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

Exemplos

$$sin(x) = sin(0) + cos(0)\,x - \frac{sin(0)}{2}x^2 - \frac{cos(0)}{3!}x^3 + \frac{sin(0)}{4!}x^4 + \frac{cos(0)}{5!}x^5 + \ldots\ldots$$

$$= x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots$$

$$cos(x) = cos(0) - sin(0)\,x - \frac{cos(0)}{2}x^2 + \frac{sin(0)}{3!}x^3 + \frac{cos(0)}{4!}x^4 - \frac{sin(0)}{5!}x^5 + \ldots\ldots$$

$$= 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots$$

$$\exp(x) = exp(0) + \exp(0)\,x + \frac{\exp(0)}{2}x^2 + \frac{exp(0)}{3!}x^3 + \frac{exp(0)}{4!}x^4 + \frac{exp(0)}{5!}x^5 + \ldots\ldots$$

$$= 1 + \frac{x^3}{3!} + \frac{x^4}{4!} + \frac{x^5}{5!} + \ldots$$

Séries de Taylor (2)

Desenvolvimento em <u>série de potências de *x-a*</u> de uma função f com derivadas contínuas

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

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

$$sin(x) = sin(\pi/2) + cos(\pi/2)\,(x-\pi/2) - \frac{sin(\pi/2)}{2}(x-\pi/2)^2 - \frac{cos(\pi/2)}{3!}(x-\pi/2)^3 + \frac{sin(\pi/2)}{4!}(x-\pi/2)^4 + \ldots$$

$$= 1 - \frac{(x-\pi/2)^2}{2!} + \frac{(x-\pi/2)^4}{4!} - \frac{(x-\pi/2)^6}{6!} \ldots$$

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

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

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

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

Majoração dos erros de truncatura (1)

$$f(x) = f(a) + f'(a)\,(x-a) + \frac{f''(a)}{2}(x-a)^2 + \frac{f'''(a)}{3!}(x-a)^3 + \ldots + \frac{f^{(k)}(a)}{k!}(x-a)^k + R_k(x)$$
$$= T_k(x) + R_k(x)$$

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

Resto na forma de Lagrange

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

θ é um ponto que está entre a e x.

Se $|f^{(k+1)}(\theta)| \leq M$, para qualquer θ 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

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

,

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 =0.375000000000000

>> abs(exp(-1)-T4)
ans =0.007120558828558

>>1/factorial(5)
ans =0.008333333333333
```

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 $ax^2 + bx + c = 0$

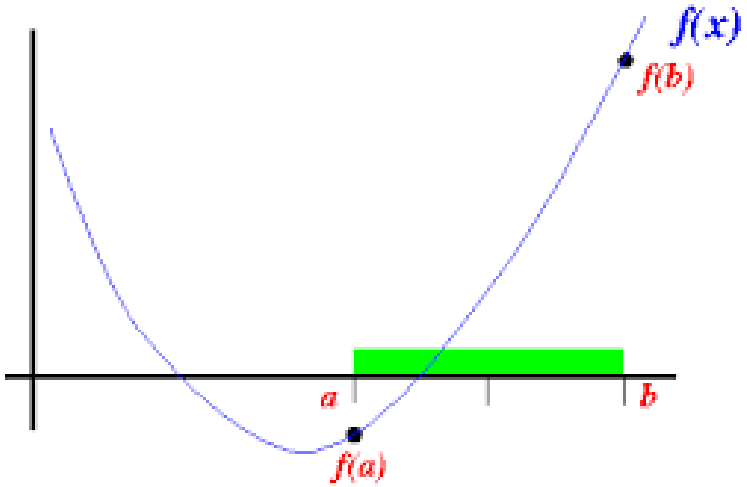
$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

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 $f(a).f(b) < 0$, então existe pelo menos uma raiz r da equação $f(x) = 0$ entre a e b.



Método da bissecção

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

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 $\mathrm{f}(m)$ então $\mathrm{r} \leftarrow m$

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

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

(ver figura 2.2 na p.42 do livro)

Método da bissecção. 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 $\mathrm{f(x)} = 0$, onde

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

Consideremos que o investidor deposita anualmente $v = 1000$ e que depois de 5 anos recebe o capital $\mathrm{M} = 6000$ 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 bissecção. 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 bissecção (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

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

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

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

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

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 bissecção (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

concluímos que k_{min}=14

nota: se efetuarmos 14 iterações obtemos a aproximação

$$x^{(14)}=0.0614\ldots$$

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

Vantagens e desvantagens do método da bissecção

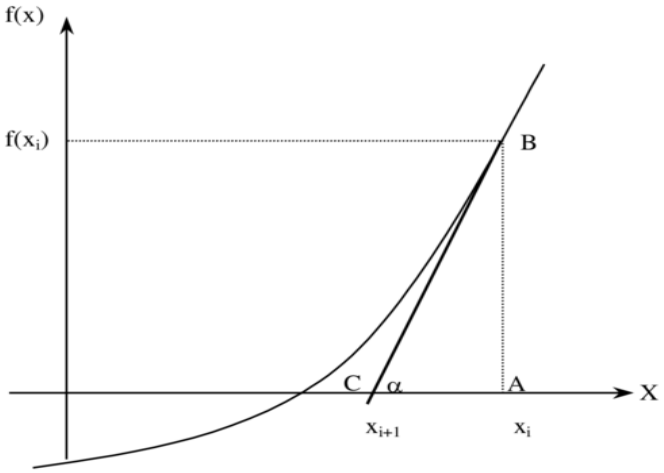
VANTAGENS:

- 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 $x^{(i)}$ interseta o eixo dos xx

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 $a = x^{(i)}$):

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

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

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

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

e

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

A convergência quadrática

Resulta

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

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

Para a iteração seguinte

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

onde μ está entre r e $x^{(i+1)}$...

Ordem de convergência

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

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

Definição: num método iterativo, se

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

- para i suficientemente grande, $r - x^{(i+1)} \approx C.\left(r - x^{(i)}\right)^p$
- p (positivo não necessariamente inteiro) é a ordem de convergência do método
- C é a constante de convergência assimptótica
- $p = 2$ 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) % 1ª iteração
x =0.118642027821101

>> x=x-f(x)/df(x) % 2ª iteração
x=**0.06**5390200813148

>> x=x-f(x)/df(x) % 3ª iteração
x=**0.0614**22972148339

>> x=x-f(x)/df(x) % 4ª iteração
x=**0.0614024**12085601

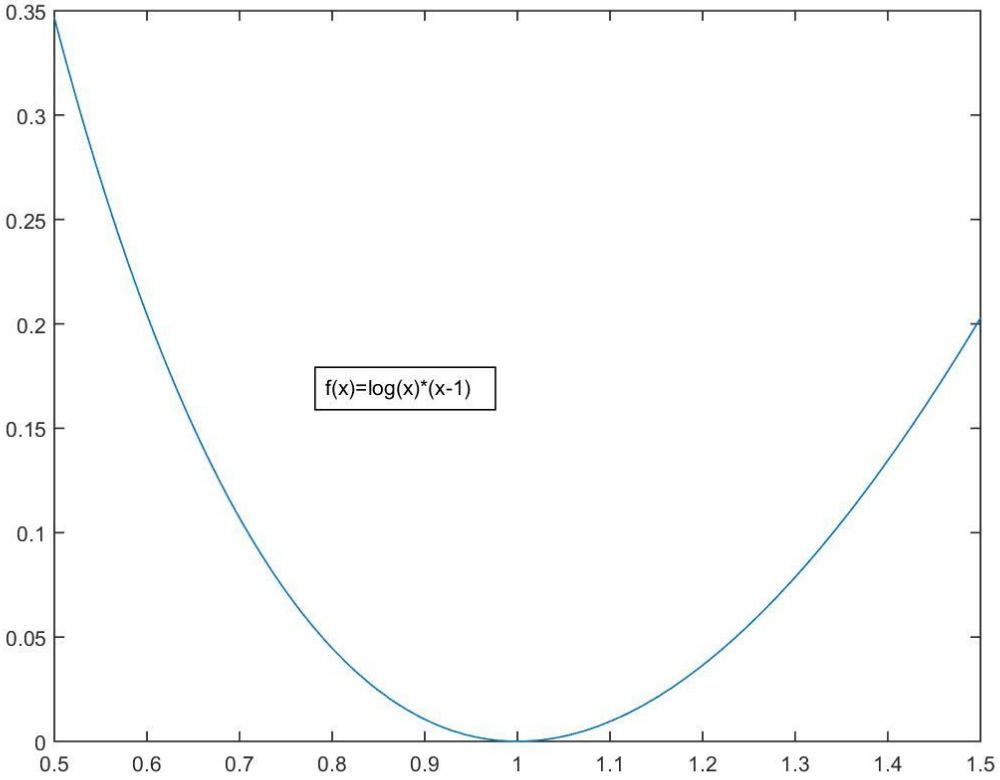
>> x=x-f(x)/df(x) % 5ª iteração
x=**0.061402411536525**

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

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

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 (ver figura 2.4 do livro)



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

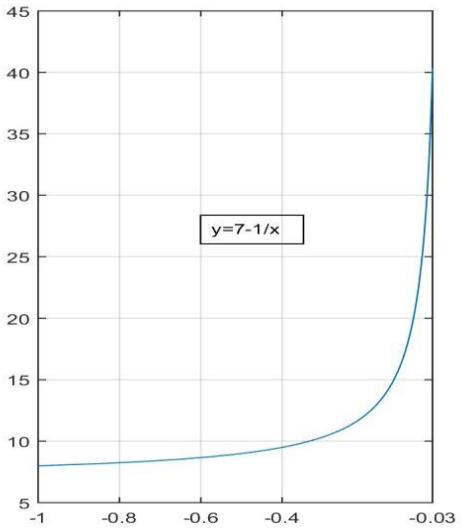
Para calcular o valor de $1/7$ (sem usar divisão), podemos começar com $x^{(0)} = 0.1$ e usar a fórmula anterior para calcular $x^{(1)}, x^{(2)}, \ldots$

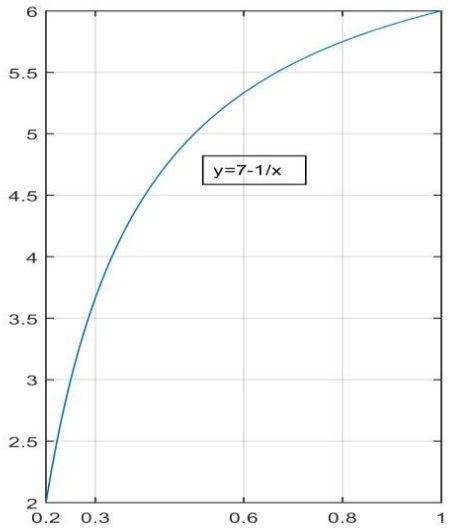
O método de Newton nem sempre converge

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

A tangente à curva no ponto de abcissa 0.3 interseta o eixo dos xx no ponto de abcissa -0.03

```
>> b=7; x=0.3;
>> x=x*(2-b*x)
x =  -0.030000000000000
```





Critérios de paragem

$x^{(0)}, x^{(1)}, x^{(2)}, \ldots x^{(i)}, \ldots \rightarrow r$ parar quando $|e^{(i)}| = \left|x^{(i)} - r\right| < tol$?

Fixada uma tolerância tol:

- $\left|x^{(i)} - x^{(i-1)}\right| < tol$ (teste sobre o incremento)
- $\left|f(x^{(i)})\right| < tol$ (teste sobre o resíduo)

O erro $\left|x^{(i)} - r\right|$ e o resíduo podem ser muito diferentes (ver figura 2.5, p.59 do livro)

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 φ é uma raiz da equação

$$x = \varphi(x)$$

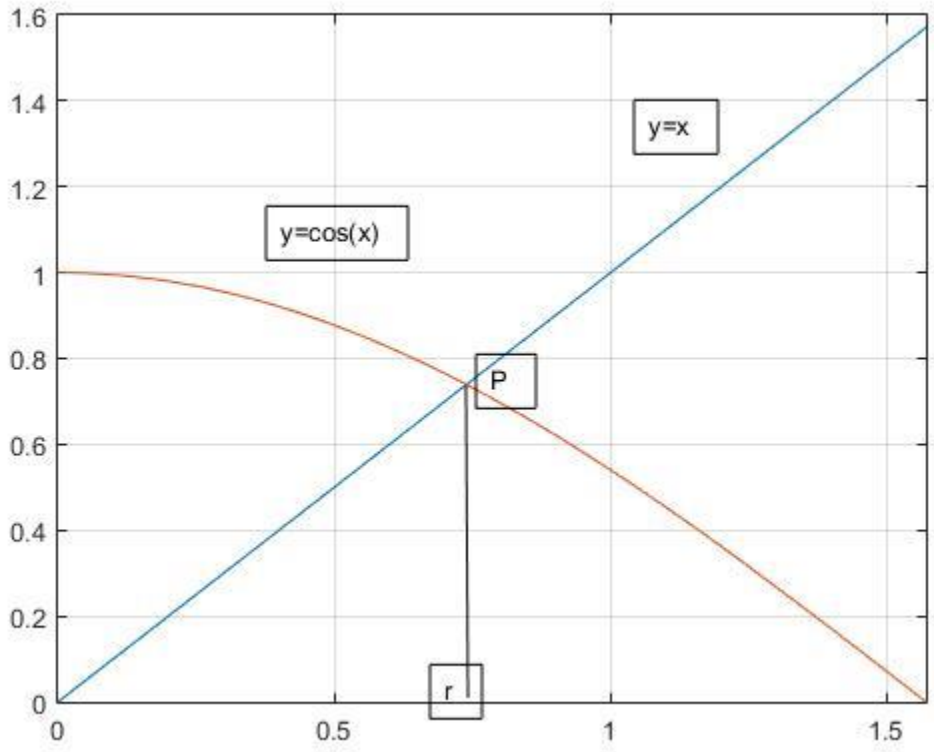
ou

$$\mathrm{f}(x) = 0$$

com $\mathrm{f}(x) = x - \varphi(x)$.

O método do ponto fixo (2)

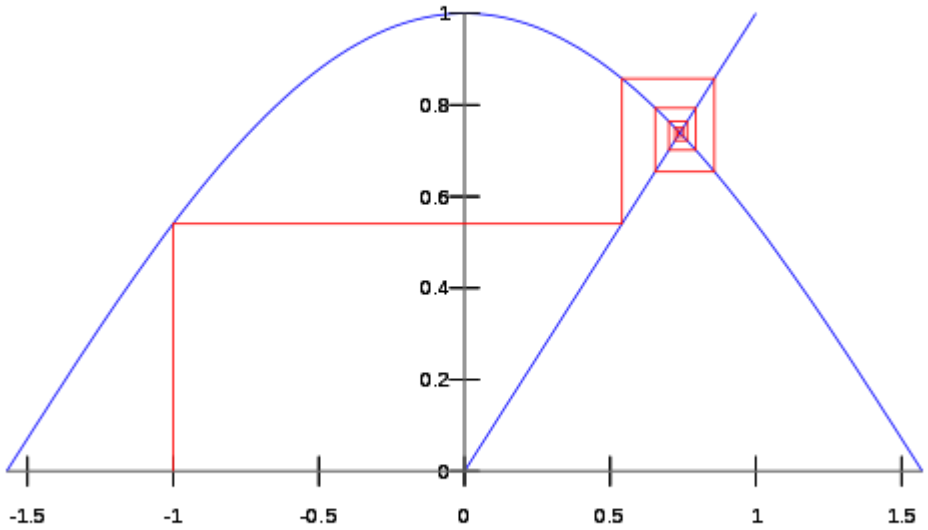
O ponto fixo de cos(x) é a abcissa do ponto P em que a curva da função coseno interseta a reta y=x.



Interpretação geométrica das iterações do ponto fixo

O ponto fixo de cos(x) é a abcissa do ponto P em que a curva da função coseno interseta a reta y=x.

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



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 φ e o valor inicial $x^{(0)}$ para que a sucessão seja convergente para o ponto fixo?

Teorema do valor médio de Lagrange: se φ tem derivada contínua em [a,b], então existe pelo menos um ponto θ 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 θ. Em pelo menos um instante, a velocidade é igual à velocidade média.

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

Análise da convergência do método do ponto fixo (2)

Aplicando o teorema do valor médio à **função iteradora φ** no intervalo $[x^{(k)}, r]$: existe θ 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

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

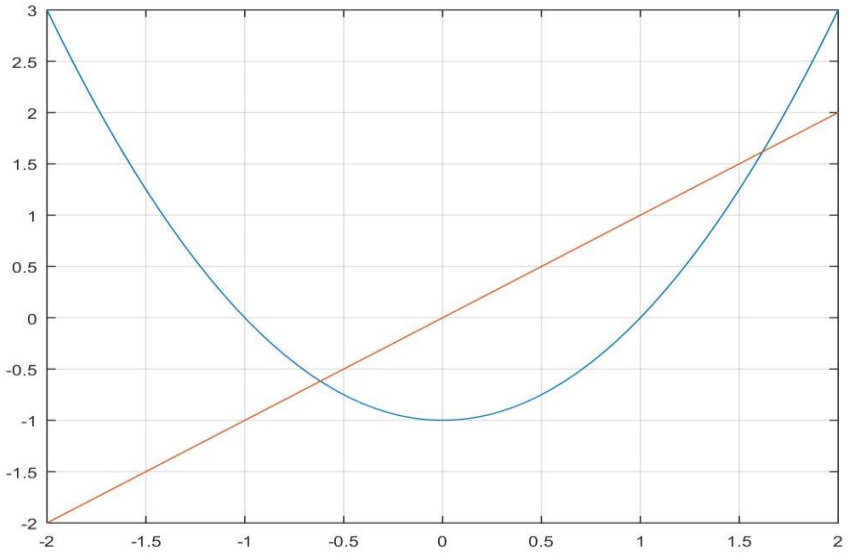
Análise da convergência do método do ponto fixo (3)

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

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

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

Exemplo 2: $\varphi(\mathrm{x}) = x^2 - 1$ tem dois pontos fixos $r_{\pm} = \frac{1\pm\sqrt{5}}{2}$ mas $|\varphi'(r_{\pm})| = \left|1 \pm \sqrt{5}\right| > 1$. 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. $x = e^{-x}$, $\varphi_1(x) = e^{-x} \Rightarrow \varphi'_1(x) = -e^{-x}$

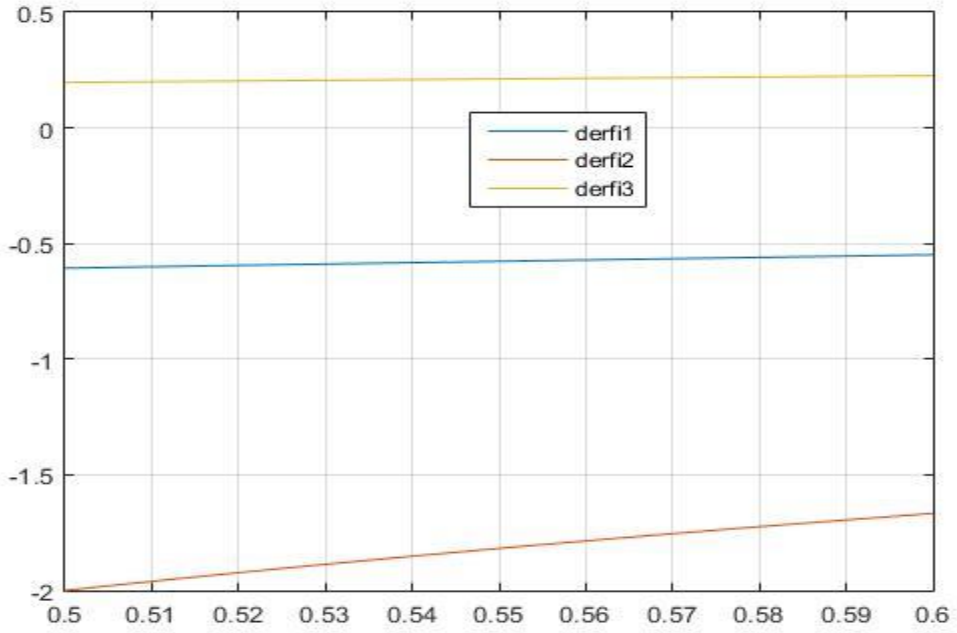
ii. $x = -\log(x)$, $\varphi_2(x) = -\log(x) \Rightarrow \varphi'_2(x) = -\frac{1}{x}$

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

Começando por observar que a raiz da equação está entre 0.5 e 0.6, no Matlab produzimos os gráficos de φ'_1, φ'_2 e φ'_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 φ_1 e φ_3 produzem sucessões convergentes desde que $0.5 < x^{(0)} < 0.6$ porque neste intervalo é $|\varphi'_1(x)|<1$ e é $|\varphi'_3(x)|<1$;
- φ_3 produz convergência mais rápida porque $|\varphi'_3(x)|<|\varphi'_1(x)|$;
- A sequência produzida com φ_2 diverge porque $|\varphi'_2(x)|>1$

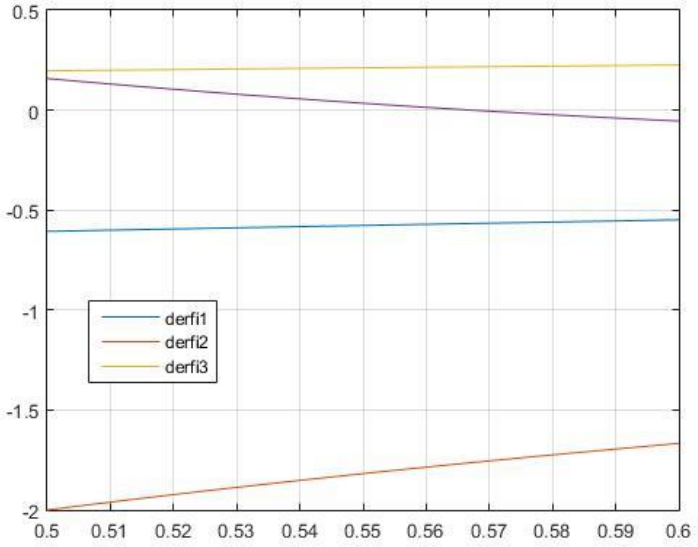
Uma escolha especial da função iteradora

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

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

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

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



>> 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 $\left|x^{(k+1)} - r\right|$ (1)

Se pararmos a as iterações quando $\left|x^{(k+1)} - x^{(k)}\right| <$ tol, podemos garantir $\left|r - x^{(k)}\right| < tol$?
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.
- $-1 < \varphi'(x) < 0 \Rightarrow \left|r - x^{(k)}\right| < |x^{(k+1)} - x^{(k)}|$
- $\varphi'(x) \approx 1 \Rightarrow \left|r - x^{(k)}\right| \gg |x^{(k+1)} - x^{(k)}|$

Estimativa do erro $\left|x^{(k+1)} - r\right|$ (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 $\frac{1}{|f'(r)|}$.

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

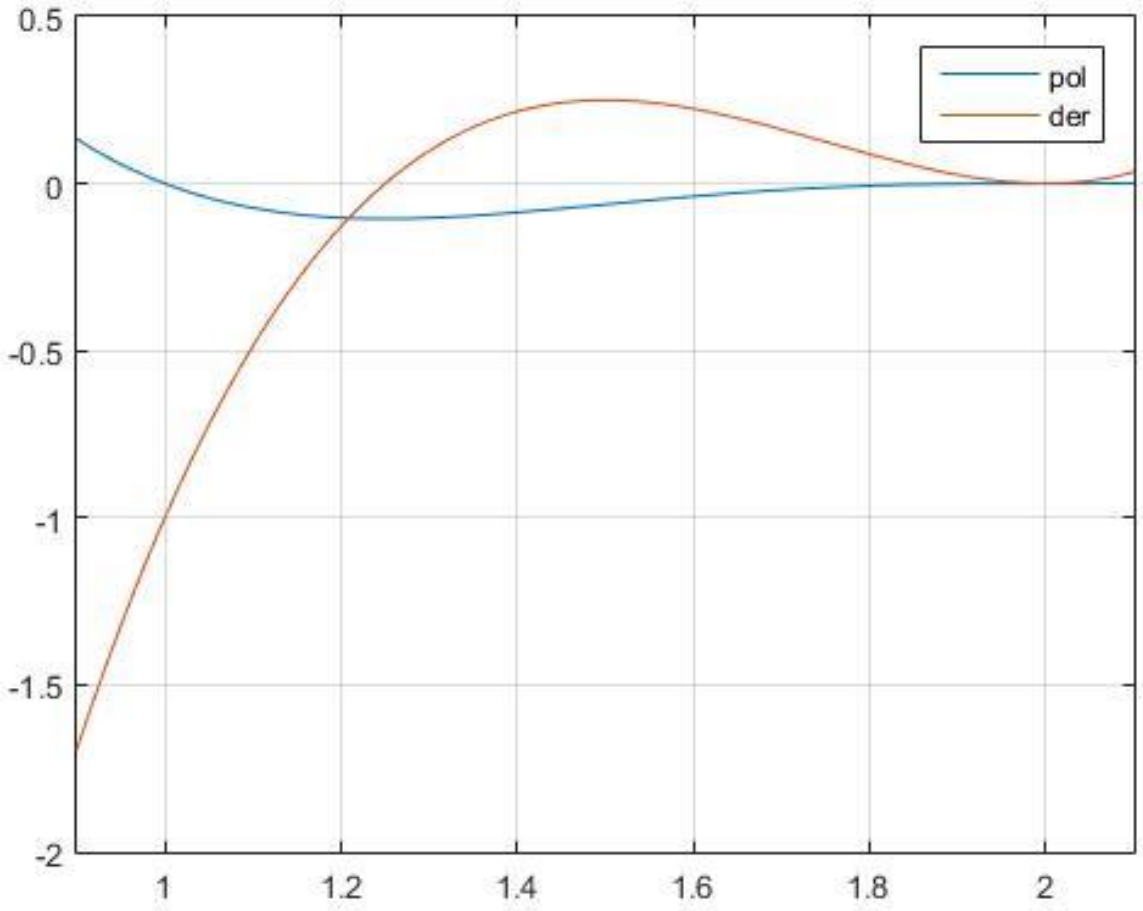
$p'(x) = 4x^3 - 21x^2 + 36x - 20$

Tem-se $p'(2.001) = 3.004e - 6$ e o erro 0.1187 é "apenas" 3 ordens de grandeza maior do que 10^{-4}, isto é, não é tão grande quanto $\frac{10^{-4}}{3*10^{-6}}$.

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

,