Matemática Computacional

Aula 1 - Aproximações numéricas

Ricardo Moura

Apresentação escrita de números reais

O número $5312,15$ representa na verdade:

$$5.10^3 + 3.10^2 + 1.10^1 + 2.10^0 + 4.10^{-1} + 5.10^{-2}$$

sendo na verdade diferente de $-5312,15$. Apenas temos um número que tem dupla representação: o zero!

$$5.31215\ E3 \rightarrow 5.31215.10^2$$

é outra possível representação deste número.
Mas há números que não têm representação finita

$$\pi \approx 3,1416$$

carregando um erro menor que o nível de precisão escolhido.

Notação posicional em base arbitrária

Definition

Seja b um número natural maior que 1 e $A_b = 0, 1, 2, ..., b-1 \subset \mathbb{N}$, diz-se que um número real x é representado em base b como

$$a_{m-1}a_{m-2}\ldots a_0.a_{-1}\ldots a_{-n} \quad (\forall a_i \in A_b)$$

escrevendo-se $x = (a_{m-1}a_{m-2}\ldots a_0.a_{-1}\ldots a_{-n})_b$ se e só se verificar

$$x = \sum_{i=-n}^{m-1} a_i b^i$$

Notação posicional em base arbitrária

Example

- $12^\circ 35' 21'' = 12 + 35.10^{-1} + 21.10^{-2}$
- $(1C)_{16} = 1.16^1 + 12.16^0 = 28$
- $((10)(54)(61))_{100} = (105461)_{10}$
- $(5632)_8 = ((101)(110)(011)(010))_8 = (101110011010)_2 = 5.8^3 + 6.8^2 + 3.8^1 + 2 = (2970)_{10}$

Notação científica em base b

Definition

Seja $x \neq 0$ um real qualquer e $b \neq 1$ uma base natural qualquer, diz-se que x tem expoente $t \in \mathbb{Z}$ se $|x| \in [b^t, b^t + 1]$. A sua representação em notação científica será

$$x = \pm(a_0.a_1 a_2 \ldots a_{p-1})_b \times b^t \implies \frac{|x|}{b^t} = (a_0.a_1 a_2 \ldots a_{p-1})_b \quad a \neq 0$$

t - expoente
$a_0 a_1 a_2 \ldots a_{p-1}$ - mantissa de comprimento p (precisão p)
b - base

O zero é o único número que não admite esta representação.

Notação científica em base b

Example

- $(5632)_8 = (5.632)_8 \times 8^3 = (2.970)_{10} \times 10^3$

  Calcular o expoente
  $t_1 = \lfloor log_8 |2970| \rfloor \approx \lfloor 3.8454 \rfloor = 3$
  $t_2 = \lfloor log_{10} |2970| \rfloor \approx \lfloor 3.7507 \rfloor = 3$

Notação científica em base b

Example

- $(5632)_8 = (5.632)_8 \times 8^3 = (2.970)_{10} \times 10^3$

  Calcular o expoente
  $t_1 = \lfloor log_8|2970| \rfloor \approx \lfloor 3.8454 \rfloor = 3$
  $t_2 = \lfloor log_{10}|2970| \rfloor \approx \lfloor 3.7507 \rfloor = 3$

- E para base 2?
  $t_3 = \lfloor log_2|2970| \rfloor \approx \lfloor 11.536 \rfloor = 11$
  Logo na base 2 o valor de x será obtido fazendo
  $a_0 = \lfloor 2970/2^{11} \rfloor = \lfloor 1.4502 \rfloor = 1;$
  $a_1 = \lfloor 2 * 0.4502 \rfloor = \lfloor 0.90040 \rfloor = 0;\ a_2 = \lfloor 2 * 0.9004 \rfloor = 1;$
  $a_3 = 1;\ a_4 = 1;\ a_5 = 0;\ a_6 = 0;\ a_7 = 1;\ a_8 = 1;\ a_9 = 0;$
  $a_{10} = 0;\ a_{11} = 1$
  $x = (101110011010)_2 = 2^{11} + 2^9 + 2^8 + 2^7 + 2^4 + 2^3 + 2^2 = 2970$
  dec2bin(number) dá o número em Octave

Notação científica em base b

Example

```
x=pi
y=1/pi^50
x
y
format long
x
y
format long e
x
y
printf('%.48f\n', x)
printf('%.48f\n', y)
printf('%.70f\n', x)
printf('%.200f\n', y)
```

Notação científica em base b

Example

Exercício: Escrever *pi* $\approx (3.1416)_{10}$ em base binária com 15 algarismos significativos. Compare com a representação binária de $(31416)_1 0$.

Resposta:

$(3.11037)_8 = (11.0010010000111111)_2$

$(75270)_8 = (111101010111000)_2$

Nota: Se um par de números pode ser representado em base *b*, também a sua soma, diferença e produto podem ser representados, no entanto, o seu quociente. Com números não inteiros, temos de sempre passar pela base 10 antes de mudarmos para outra base.

Representação de números negativos

1.º método - Reservar um bit para marcar o **sinal** do número.
2.º método - **Complemento do b** em base b (b par).

$$x \in [0, b^q/2 - 1] \cap \mathbb{Z} \mapsto x \in [0, b^q/2 - 1] \cap \mathbb{Z}$$

$$-x \in [-b^q/2, -1] \cap \mathbb{Z} \mapsto b^q - x \in [b^q/2, b^q - 1] \cap \mathbb{Z}$$

$-b^q/2$ não tem o seu correspondente negativo
3.º método - **Complemento do** $\mathbf{b-1}$.

$$x \in [0, b^q/2 - 1] \cap \mathbb{Z} \mapsto x \in [0, b^q/2 - 1] \cap \mathbb{Z}$$

$$b^q - 1 - x \in [-b^q/2 + 1, 0] \cap \mathbb{Z} \mapsto -x \in [b^q/2, b^q - 1] \cap \mathbb{Z}$$

O zero tem duas representações $0 \ldots 0$ e $(b^q - 1) \ldots (b^q - 1)$

Representação de números negativos

4.º método - **Excesso em $\mathbf{b^q/2}$.**

$$x \in [-b^q/2, b^q/2 - 1] \cap \mathbb{Z} \mapsto x + b^q/2 \in [0, b^q - 1] \cap \mathbb{Z}$$

$-b^q/2$ não tem o seu correspondente negativo

5.º método - **Excesso em $\mathbf{b^q/2 - 1}$.**

$$x \in [-b^q/2 + 1, b^q/2] \cap \mathbb{Z} \mapsto x + b^q/2 - 1 \in [0, b^q - 1] \cap \mathbb{Z}$$

$-b^q/2 + 1$ não tem o seu correspondente negativo

Representação de números negativos

Example

Em palavras de 2 bytes ($q = 16$ bits binários) $x = -16521$ e $y = -320$

- **1**100000010001001 e **1**000000101000000

Representação de números negativos

Example

Em palavras de 2 bytes ($q = 16$ bits binários) $x = -16521$ e $y = -320$

- **1**100000010001001 e **1**000000101000000
- De $2^{16} - 16521 = 49015$ e $2^{16} - 320 = 65216$ vem que 1011111101110111 e 1111111011000000

Representação de números negativos

Example

Em palavras de 2 bytes ($q = 16$ bits binários) $x = -16521$ e $y = -320$

- **1**100000010001001 e **1**000000101000000
- De $2^{16} - 16521 = 49015$ e $2^{16} - 320 = 65216$ vem que 1011111101110111 e 1111111011000000
- De $2^{16} - 1 - 16521 = 49014$ e $2^{16} - 1 - 320 = 65216$ vem que 1011111101110110 e 1111111010111111

Representação de números negativos

Example

Em palavras de 2 bytes ($q = 16$ bits binários) $x = -16521$ e $y = -320$

- **1**100000010001001 e **1**000000101000000
- De $2^{16} - 16521 = 49015$ e $2^{16} - 320 = 65216$ vem que 1011111101110111 e 1111111011000000
- De $2^{16} - 1 - 16521 = 49014$ e $2^{16} - 1 - 320 = 65216$ vem que 1011111101110110 e 1111111010111111
- 0011111101110111 e 0111111011000000

Representação de números negativos

Example

Em palavras de 2 bytes ($q = 16$ bits binários) $x = -16521$ e $y = -320$

- **1**100000010001001 e **1**000000101000000
- De $2^{16} - 16521 = 49015$ e $2^{16} - 320 = 65216$ vem que 1011111101110111 e 1111111011000000
- De $2^{16} - 1 - 16521 = 49014$ e $2^{16} - 1 - 320 = 65216$ vem que 1011111101110110 e 1111111010111111
- 0011111101110111 e 0111111011000000
- 0011111101110110 e 0111111010111111

Ponto flutuante

Considere-se a notação científica

$$\pm a_0.a_1a_2\ldots a_{p-1}\times b^t;\quad a_i\in\{0,1,\cdots-1\}\wedge a_0\neq 0$$

Ponto flutuante

Considere-se a notação científica

$$\pm a_0.a_1a_2\ldots a_{p-1}\times b^t;\quad a_i\in\{0,1,\cdots-1\}\wedge a_0\neq 0$$

É necessário guardar:
- o sinal;
- a mantissa;
- e o expoente t.

Ponto flutuante

Considere-se a notação científica

$$\pm a_0.a_1a_2\ldots a_{p-1}\times b^t;\quad a_i\in\{0,1,\cdots-1\}\wedge a_0\neq 0$$

É necessário guardar:

- o sinal;

- a mantissa;

- e o expoente t.

Na representação numérica binária com ponto flutuante utiliza-se:

1. 1 bit para o sinal;

2. vários bits para o expoente;

3. vários bits para a mantissa (sempre positiva).

Ponto flutuante

- Primeiro bit 1 se negativo e 0 se positivo;

Ponto flutuante

- Primeiro bit 1 se negativo e 0 se positivo;
- Representação do número em notação científica em base 2, **normalizado** onde $a_0 = 1$ e uma mantissa de p algarismos.

Ponto flutuante

- Primeiro bit 1 se negativo e 0 se positivo;
- Representação do número em notação científica em base 2, **normalizado** onde $a_0 = 1$ e uma mantissa de p algarismos.
- Nos bits seguintes regista o expoente através do método do excesso de $2^{q-1} - 1$.

Ponto flutuante

- Primeiro bit 1 se negativo e 0 se positivo;
- Representação do número em notação científica em base 2, **normalizado** onde $a_0 = 1$ e uma mantissa de p algarismos.
- Nos bits seguintes regista o expoente através do método do excesso de $2^{q-1} - 1$.
- Nos últimos bits regista a mantissa binária, tendo usualmente $p - 1$ posições pois o primeiro já se sabe que é sempre 1.

Ponto flutuante

- Primeiro bit 1 se negativo e 0 se positivo;
- Representação do número em notação científica em base 2, **normalizado** onde $a_0 = 1$ e uma mantissa de p algarismos.
- Nos bits seguintes regista o expoente através do método do excesso de $2^{q-1} - 1$.
- Nos últimos bits regista a mantissa binária, tendo usualmente $p - 1$ posições pois o primeiro já se sabe que é sempre 1.

Example

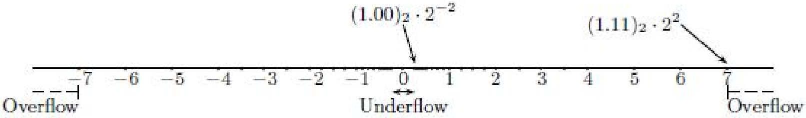
$x = -7$

1. tem sinal 1
2. not. científica 1.11×2^2
3. Binário de $2^{10} - 1 + 2 = (10000000001)_2$
4. $(1100000000\ldots)$

$$1 \quad 10000000001 \quad 110000000000\ldots0000000000000000$$

Ponto flutuante

- Representação finita
- conjunto de números representáveis é mais denso na viz. de 0.
- Existe um número positivo máximo onde números podem ser escritos exatamente (os outros estão na zona overflow) e um positivo mínimo (números entre este e 0 só se representam aproximadamente zona underflow)



Erro absoluto e erro relativo

Considere-se x^* a aproximação do valor exato x.

Definition

Chamamos **erro absoluto** a

$$E(x^*, x) = |x^* - x|$$

e **erro relativo** a

$$\delta(x^*, x) = \frac{|x^* - x|}{|x|}, x \neq 0.$$

Nota1: Quando falamos de "erro" estamos a referir-nos ao erro absoluto ou a $x^* - x$.

Nota2: O erro relativo também pode ser expresso como $\delta_{x^*} = \frac{|x^* - x|}{|x^*|}$

Erro absoluto e erro relativo

Será utilizado com frequência a notação $x^* \pm \epsilon$ como a aproximação de x, que irá significar que

$$\exists e \in [-\epsilon, \epsilon] : x^* = x + e \implies |x^* - x| \leq \epsilon.$$

Erro absoluto e erro relativo

Será utilizado com frequência a notação $x^* \pm \epsilon$ como a aproximação de x, que irá significar que

$$\exists e \in [-\epsilon, \epsilon] : x^* = x + e \implies |x^* - x| \leq \epsilon.$$

Também poderemos escrever $x = x^*(1 \pm \delta)$

Erro absoluto e erro relativo

Será utilizado com frequência a notação $x^* \pm \epsilon$ como a aproximação de x, que irá significar que

$$\exists e \in [-\epsilon, \epsilon] : x^* = x + e \implies |x^* - x| \leq \epsilon.$$

Também poderemos escrever $x = x^*(1 \pm \delta)$

Definition

Considerando $b^e \leq x \leq b^{e+1}$ diz-se que x^* é uma aproximação de x **com** p **algarismos significativos** se:

$$\left|\frac{x^* - x}{b^e}\right| \leq \frac{b}{2} \times b^{-p}, \textbf{para b par}$$

Example

0.008234 ± 0.00004 aproximação com 3 algarismos significativos
0.008234 ± 0.00006 aproximação com 2 algarismos significativos,

Erro absoluto e erro relativo

Definition

Seja x um valor real, podemos escrevê-lo normalizado em base b através de:

$$x = \pm((a_0.a_1 \dots a_{p-1})_b \times b^e + \epsilon),$$

onde $a_0 \neq 0$ e $0 \leq \epsilon < (0.0 \dots \overbrace{1}^{p-1})_b \times b^e = b^{e-p+1}$.

O valor $x^* = \pm(a_0.a_1 \dots a_{p-1})_b \times b^e$ é a aproximação de x por **arredondamento por corte com p algarismos**.

Example

$x = (1.0111011....)_2$ arredondado por corte com 2 algarismos seria $x^* = (1.0)_2$, mas no entanto, x está na verdade mais próximo de $(1.1)_2$.

Erro absoluto e erro relativo

Arredondamento ao mais próximo com p algarismos - arredondamento simétrico

1. Para $\epsilon < b/2 \times b^{e-p}$, então o arredondamento simétrico será $x^* = \pm(a_0.a_1 \dots a_{p-1}) \times b^e$.
2. Para $\epsilon > b/2 \times b^{e-p}$, então o arredondamento simétrico será $x^* = \pm\left((a_0.a_1 \dots a_{p-1}) + b^{-(p-1)}\right) \times b^e$.
3. Para $\epsilon = (b/2) \times b^{e-p}$ procede-se por hábito proceder como em 1) para a_{p-1} par e como em 2) para a_{p-1} ímpar.

Example

Arredondamentos em base decimal $0.2397 \approx 0.240$; $0.2392 \approx 0.239$; $0.2375 \approx 0.238$; $0.2365 \approx 0.236$.
Arredondamentos em base binária $10.11100011 \approx 10.1110$; $0.00100100011 \approx 0.00100100$; $1.111111111110 \approx 10.0000$; $110.101101 \approx 110.110$.

Erro absoluto e erro relativo

Proposition

Para qualquer número $x \in \mathbb{R}$ *tal que* $|x| \in [b^e, b^{e-1}[$ *há uma correspondente aproximação* x^* *em ponto flutuante tal que:*

- $\delta(x^*, x) \leq (1/2)b^{-p+1}$ *quando* x^* *é obtido por arredondamento simétrico (*p *algarismos significativos).*
- $\delta(x^*, x) < b^{-p+1}$ *quando* x^* *é obtido por arredondamento por corte (*p *algarismos significativos).*

Nota: O valor de x não é *overflow*/*underflow*.

Erro absoluto e erro relativo

Definition

Unidade de arredondamento duma máquina é o menor valor de u tal que:

$$|\delta(x^*, x)| < u$$

para todos os arredondamentos x que não sejam *overflow*/*underflow*.

Definition

Épsilon da máquina numa máquina que usa computações em ponto flutuante é o menor ϵ que verifica:

$$(1 + \epsilon)* > 1$$

onde $(1 + \epsilon)*$ representa a soma tal como seria feita pela máquina para números em ponto flutuante.

Análise do erro

A raiz mais alta de $x^2/2 + ax + 5 = 0$ pode ser resolvido através de

$$f : a \mapsto -a + \sqrt{a^2 - 10}$$

Pode-se criar um método numérico

$$f^* : a^* \mapsto f^*(a^*)$$

.

Ter $f^*(a^*)$ como aproximação de $f(a)$ contem um erro

$$\pm E(f^*(a^*), f(a)) = \pm E(f^*(a^*), f(a^*)) \pm E(f(a^*), f(a)),$$

isto é, um **erro de computação**, o primeiro da expressão, e um **erro propagado**, o segundo.

Propagação do erro

Definition

Diz-se que um problema resolvido por uma função $f : \mathbb{R} \to \mathbb{R}$ está bem condicionado se a variações pequenas nos dados introduzidos levar a variações pequenas nos resultados obtidos.

Ponto de vista de erro absoluto:

$$|a^* - a|\text{pequeno} \implies |f(a^*) - f(a)| pequeno$$

para valores pequenos de $|a^* - a|$ a noção de pequeno é medida pelo valor de $\bar{k}$ que satisfaz:

$$|f(a^*) - f(a)| < \bar{k}|a^* - a|.$$

Propagação do erro

Definition

Diz-se que um problema resolvido por uma função $f : \mathbb{R} \to \mathbb{R}$ está bem condicionado se a variações pequenas nos dados introduzidos levar a variações pequenas nos resultados obtidos.

Ponto de vista de erro relativo:

$$\frac{|a^* - a|}{|a|}\text{pequeno} \implies \frac{|f(a^*) - f(a)|}{|f(a)|}\text{pequeno}$$

para valores pequenos de $|a^* - a|$ a noção de pequeno é medida pelo valor de k que satisfaz:

$$\frac{|f(a^*) - f(a)|}{|f(a)|} < k\frac{|a^* - a|}{|a|}.$$

Propagação do erro

Proposition

Se f é uma função diferenciável que resolve o problema, e se $a^* = a(1 \pm \delta)$, então o erro relativo entre $f(a^*)$ e $f(a)$ depende de δ na forma:

$$\delta(f(a^*), f(a)) = \left| \frac{a.f'(a)}{f(a)} \right| .\delta + \mathcal{O}(\delta^2)$$

e, se $a^* = a + \epsilon$, então o erro absoluto entre $f(a^*)$ e $f(a)$ depende de ϵ na forma:

$$E(f(a^*), f(a)) = \left| f'(a) \right| .\epsilon + \mathcal{O}(\epsilon^2).$$

Propagação do erro

Definition

Ao valor de

$$k = \left| \frac{a.f'(a)}{f(a)} \right|$$

chamamos **número de condição** da função f no ponto a.

Quando temos um valor aproximado a^* de a, a solução fornecida por f pode ter um erro relativo até $k.\delta$. O problema diz-se mal condicionado se o valor de k é grande.

Propagação do erro

Example

Para valores de a longe de $\sqrt{10}$, temos k pequeno

$$k = \left| \frac{a.f'(a)}{f(a)} \right| = \left| \frac{-a}{\sqrt{a^2 - 10}} \right| \approx 1.$$

Mas para valores na vizinhança de $\sqrt{(10)}$, o problema está mal condicionado. Experimentar $a = 3.16229$;$a = 3.16228$ e $a = 3.16227$.

```
function f = raiz(x)
f=-x+sqrt(x^2-10)
end
```

Análise de erros do algoritmo

Analisar o erro associado ao algoritmo numérico:

$$E(f^*(a^*), f(a^*))$$

Definition

Diz-se que um algoritmo é estável se erros pequenos nos dados introduzidos levam a erros pequenos nos resultados numéricos obtidos através do algoritmo.

Esta componente é uma fonte de erros e é necessário limitar os mesmos, para isso, pode-se fazer uma análise de erros direta ou uma análise de erros inversa.

Análise de erros do algoritmo

Análise de erros direta

Tendo em conta

$$\delta(f^*(a^*), f(a^*)) = \left| \frac{f^*(a^*) - f(a^*)}{f(a^*)} \right|,$$

um algoritmo será melhor se este valor for sempre menor do que a unidade de arredondamento u.
Se $f^*(a^*) = f(a^*).(1+\delta), \delta < u$, então a substituição de f por f^* não acrescenta erro nenhum.

Análise de erros do algoritmo

Análise de erros inversa

Neste ponto de vista, a pergunta que se faz não é se $f^*(a^*)$ é aproximadamente igual a $f(a^*)$, mas se $f^*(a^*)$ é exatamente igual a $f(a)$ para algum a, tal que $a = a^*(1 + \delta)$ com $\delta < u$.

Análise de erros do algoritmo

Análise de erros inversa

Example

Considerando outra vez $x^2/2 + ax + 5$, podemos encontrar a raiz deste polinómio através do polinómio de Taylor, para cada a^*, centrado em $11/2$ e truncado no terceiro passo,

$$f^*(a^*) = \frac{2.(20(a^*)^2 - 139a^* - 205)}{729}$$

O erro cometido pelo algoritmo em $a^* = 5.4$, pelo método direto é dado por

$$\delta = \delta(f^*(5.4), f(5.4)) = \left| \frac{\frac{2.(20\times(5.4)^2 - 139\times 5.4 - 205)}{729} - raiz(5.4)}{raiz(5.4)} \right|$$

$$= 1.08796 \times 10^{-3}$$

Análise de erros do algoritmo

Análise de erros inversa

Example

Através do método inverso, temos de encontrar um a tal que $f^*(5.4) = f(a)$

```
function f=new(x)
f=raiz(x)-2*(20*5.4^2-139*5.4-205)/729
end
fsolve(@new,5.4)
```

que será $f^*(5.4) = f(5.40476802)$, sendo o erro relativo para este método $\delta(a^*, a) = 8.82967 \times 10^{-4}$

Se δ for menor que u, neste método, pode-se dizer que não há erro acrescentado pelo algoritmo e f^* produz a solução exata (ou seja, o sistema não distingue a^* de a se $\delta(a^*, a) < u$).

Análise de erros do algoritmo

Se $\delta < u$ ou δ é duma ordem próxima a u, diz-se que o algoritmo é estável.

Definition

Um algoritmo f^* que aproxima f diz-se estável em a^* para análise de erros direta se:

$$\delta(f^*(a^*), f(a^*)) < c_1 \times u$$

para c_1 constante não demasiado grande. Um algoritmo f^* que aproxima f diz-se estável em a^* para análise de erros inversa se existir um a com $f(a) = f^*(a^*)$ e tal que :

$$\delta(a^*, a) < c_2 \times u$$

para c_2 constante não demasiado grande.

Erros das operações aritméticas

Erro propagado

$$E(soma(u^*, v^*), soma(u, v)) \leq \epsilon_u + \epsilon_v + \mathcal{O}(\epsilon^2)$$

$$E(subtr(u^*, v^*), subtr(u, v)) \leq \epsilon_u + \epsilon_v + \mathcal{O}(\epsilon^2)$$

$$E(prod(u^*, v^*), prod(u, v)) \leq |u^*|.\epsilon_v + |v^*|\epsilon_u + \mathcal{O}(\epsilon^2)$$

$$E(div(u^*, v^*), div(u, v)) \leq \frac{|u^*|.\epsilon_v + |v^*|\epsilon_u}{|v^*|^2} + \mathcal{O}(\epsilon^2)$$

para $u^* = u \pm \epsilon_u$ e $v^* = v \pm \epsilon_v$.

Erros das operações aritméticas

Da mesma forma, os erros relativos serão

$$\delta(soma(u^*, v^*), soma(u, v)) = \delta_u + \delta_v + \mathcal{O}(\delta^2)$$

$$\delta(subtr(u^*, v^*), subtr(u, v)) = \text{depende fortemente de} \quad u - v$$

$$\delta(prod(u^*, v^*), prod(u, v)) = \delta_u + \delta_v + \mathcal{O}(\delta^2)$$

$$\delta(div(u^*, v^*), div(u, v)) = \delta_u + \delta_v + \mathcal{O}(\delta^2) + \mathcal{O}(\delta^2)$$

para $u^* = u(1 \pm \delta_u)$ e $v^* = v(1 \pm \delta_v)$.