Erros numéricos

por
Mílton Procópio de Borba

1. Alguns problemas ao fazermos contas no computador

Os problemas a seguir foram analisados num Pentium, com a ajuda de pequenos programas feitos em QBasic. As listagens dos programas e os resultados aparecem no final, como anexos.

1.1 Cálculo do π

Se dobrarmos a quantidade *n* de lados de um polígono regular inscrito numa circunferência de raio unitário, o seu lado deixará de medir L_k para medir $L_{k+1} = \sqrt{2-\sqrt{4-L_k^2}}$.

0 perímetro $2p_k$ deste polígono mede $n.L_k < 2\pi$, mas com $k \to \infty$, teremos que $n.L_k \to 2\pi$.

Assim, podemos considerar que $\boxed{\pi = \lim_{k\to\infty} \frac{n.L_k}{2}}$.

Com *n* = *6*, temos $L_0 = 1$ e $p_0 = 6.1/2 = 3$. Com n = 12, temos $p_1 = 12.L_1/2 = 6.\sqrt{2-\sqrt{3}} \approx 3{,}1058$.

Em geral, para $k > 0$, temos que $p_k = 2^k.3.L_k = 2^k.3.\sqrt{2-\sqrt{4-L_{k-1}^2}}$.

Isto dá origem à seqüência: 2.2.2. ... .2.3.Raiz(2-Raiz(2+Raiz(2+ ... Raiz(2+Raiz(3))...))).
Calculando esta seqüência, p_{10} já registra um bom valor para π. Continuando, com 11 decimais, desde p_{11} até p_{22} são registrados os mesmos valores, depois os seguintes valores de p_k vão se degenerando até que de p_{32} em diante, temos valores zero (ver Anexos 1 e 2).

1.2 Derivada da Função *ln* usando diferenças finitas

Por definição, a derivada da função *ln* (logaritmo natural) num ponto x é dada por:
$f'(x) = \lim_{h\to 0} \frac{ln(x+h) - ln(x)}{h}$, que sabemos valer 1/x.

Foi calculado este quociente com *h = dx* variando de 0.1 até 10^{-20}, obtendo resultados cada vez melhores (com 7 decimais), quando *h* decrescia até 10^{-6}. Para *h* menores, os resultados foram piores e terminaram em zero (ver Anexo 3).

1.3 A soma parcial da série geométrica

É bem conhecido que a série $\sum_{n=1}^{\infty} \frac{1}{3^n}$ converge para 1/2. Como $3^{-21} \approx 2{,}868 \times 10^{-10}$, queremos somar as 20 primeiras parcelas e, com 10 dígitos, conseguir uma resposta bem perto de 0,5.
Obtivemos que $1/3 + 1/9 + ... + 3^{-20} = 0{,}4999999702$, enquanto que a soma em ordem contrária, $3^{-20} + 3^{-19} + ... + 1/3 = 0{,}50000000000$ (ver Anexo 4).

1.4 Somas sucessivas: 10.000 x 0.0001 = 0.0001 + 0.0001 + 0.0001 + ... + 0.0001

Esta soma, de 10.000 parcelas iguais a 0,0001, resultou em 1,000054. Erro de 0,0053%
(ver Anexo 5).

1.5 Cálculo da raiz quadrada de 1,69.

A equação $x^2 - 1{,}69 = 0$ foi resolvida por um processo iterativo linear combinado com o método da bisseção. Para isto, escrevemos a equação na forma x = 1,69/x, partimos de um valor aproximado $x_0 = 2$ e fazemos, $x_1 = 1{,}69/x_0$, obtendo $x_1 = 0{,}845$. Se $x_0 = x_1$ então teríamos obtido a raiz de 1,69. Como isto não ocorreu, tiramos a média entre estes valores, obtendo 1,4225. Reiniciamos o processo, agora com $x_0 = 1{,}4225$, na esperança de chegarmos ao valor 1,3. Isto não acontece e fica oscilando acima e abaixo da resposta (ver Anexo 6).

2. 0 sistema de Numeração do Computador

0 computador usa o sistema de numeração binária, com auxílio dos algarismos **0** e **1**, baseados nas potências de 2. Assim, o número inteiro **13** será representado por **1101** = **1**x2^3 + **1**x2^2 + **0**x2^1 + **1**x2^0, enquanto que a fração decimal **0,13** será **0,0010000101000111101011**10000101000111101011... =

= $2^{-3} + 2^{-8} + 2^{-10} + 2^{-14} + 2^{-15} + 2^{-16} + 2^{-17} + 2^{-19} + 2^{-21} + 2^{-22}$ + $2^{-23} + 2^{-28}$ + ... =

= 2^{-2} x ($2^{-1} + 2^{-6} + 2^{-8} + 2^{-12} + 2^{-13} + 2^{-14} + 2^{-15} + 2^{-17} + 2^{-19} + 2^{-20} + 2^{-21} + 2^{-26}$ + ...) = 2^{-2} x 0,52 =

= 2^{-2} x **0,10000101000111101011**10000101000111101011..., normalizado.

O sinal (...) significam que a representação continua e o sublinhado indica a parte periódica.

O número real -13,13 será:
-13,13 = - ($2^3 + 2^2 + 2^{0\,+}$ $2^{-3} + 2^{-8} + 2^{-10} + 2^{-14} + 2^{-15} + 2^{-16} + 2^{-17} + 2^{-19} + 2^{-21} + 2^{-22}$ + $2^{-23} + 2^{-28}$ + ...) =

= -**1101,0010000101000111101011**10000101000111101011... =

= -2^4 x **0,11010010000101000111101011**10000101000111101011..., normalizado.

3. Aritmética inteira

0 computador armazena cada número inteiro num conjunto de, por exemplo 32 bits, chamado de palavra composta por 32 algarismos **0** ou **1**, assim situados:

sinal	2^{30}	2^{29}	2^{28}	2^{27}	...	2^5	2^4	2^3	2^2	2^1	2^0
0	0	0	0	0	...	0	0	**1**	**1**	**0**	**1**
bit 0	bit 1	bit 2	bit 3	bit 4	...	bit 26	bit 27	bit 28	bit 29	bit 30	bit 31

Nesta ilustração aparece o número binário inteiro 1101 (= 13 em decimal), enquanto que

sinal	2^{30}	2^{29}	2^{28}	2^{27}	...	2^5	2^4	2^3	2^2	2^1	2^0
1	0	0	0	0	...	0	**1**	**1**	**0**	**1**	**0**
bit 0	bit 1	bit 2	bit 3	bit 4	...	bit 26	bit 27	bit 28	bit 29	bit 30	bit 31

representa o binário -**11010** (= -26 em decimal).

0 menor número inteiro que pode ser armazenado será:
111111111 ... 11111 = - ($2^{30} + 2^{29}$ + ... + $2^2 + 2^1 + 2^0$) = -(2^{31}-1) = -2.147.483.647.

Num computador com palavras de 16 bits por exemplo, o maior número inteiro seria:
01111 ... 1111 = ($2^{14} + 2^{13}$ + ... + $2^2 + 2^1 + 2^0$) = 2^{15} - 1 = 32.767.

4. Aritmética real (ponto flutuante)

0 computador armazena cada número real na forma $\boxed{s \times 2^{car} \times mant}$. Para isto, usa um conjunto de, por exemplo 32 bits, separados em dois grupos: um do bit 1 ao bit 7, para guardar a potência (**car**acterística) e outro do bit 8 ao último bit 31, para guardar a fração do número binário (**mant**issa). O bit 0 continua sendo para guardar o sinal (s) do número.

sinal núm	**sinal car**	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	...
0	**1**	0	0	0	0	**1**	**0**	1	0	0	...
bit 0	bit 1	bit 2	bit 3	bit 4	bit 5	bit 6	bit 7	bit 8	bit 9	bit 10	...

Nesta ilustração aparece a característica **1**0000**10** = **-2**, enquanto que

...	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	...	2^{-22}	2^{-23}	2^{-24}
...	0	**1**	**0**	**0**	**0**	**0**	**1**	...	**0**	**0**	**0**
...	bit 7	bit 8	bit 9	bit 10	bit 11	bit 12	bit 13	...	bit 29	bit 30	bit 31

representa o binário 0,**10000101000111101011000**, primeiras 24 binárias fracionárias de 0,52.

Trata-se, portanto, do número: 2^{-2} x ($2^{-1} + 2^{-6} + 2^{-8} + 2^{-12} + 2^{-13} + 2^{-14} + 2^{-15} + 2^{-17} + 2^{-19} + 2^{-20} + 2^{-21}$) ≈ 0,13.
0 maior número representável nesta palavra é quase $2^{63} \approx 10^{18}$, pois a car máxima é 2^6 - 1 = 63.

5. Erro de Arredondamento

0 erro absoluto cometido ao desprezar 2^{-25} não passa de 2^{car} x 2^{-25}. Se mant é não nula, dividindo este erro pelo número $\boxed{2^{car} \text{ x mat}}$, obtemos um erro relativo de 2^{-25}/mant.
Considerando que mant ≥ 1/2 e multiplicando por 100 (para expressar em %), concluímos que o erro percentual de arredondamento da máquina com palavras de 32 bits não passa de 100 x 2^{-24}.
Ou seja, Erro ≤ 0,00000596%
Lança-se mão do artifício de usar DUAS palavras de 32 bits, por exemplo, para obter números com maior precisão, Assim, a característica teria 56 bits (= 24+32). Este artifício é denominado de DUPLA PRECISÃO, mas na verdade a precisão é bem maior que o dobro.

6. Erros de operações matemáticas no computador

6.1 Soma (ou subtração) de número muito pequeno com número muito grande

Se somarmos ou subtrairmos um número muito pequeno (x, de característica pequena) a outro muito grande (Y, de característica grande), certamente se perderão os últimos (ou todos) os bits da mantissa de x.
Esta perda é tanto maior quanto maior a diferença entre as características de x e de Y.
Se analisarmos o problema 1.3, vemos que as últimas parcelas (3^{-15} em diante, por exemplo), são muito pequenas em relação à soma que já estava perto de 0,5.
Somando na ordem inversa, cada soma era feita entre dois número próximos.
Realmente, $3^{-20} + 3^{-19} \approx 3^{-18}$ e Sorna - 1/3 ≈ 1/3.

6.2 Subtração de dois números quase iguais

Se subtrairmos dois números (x e y) quase iguais com vários algarismos significativos, o resultado pode ser um número com poucos algarismos significativos.
A diferença significativa estaria nos algarismos (bits) desprezados ao representar x e y.
Este fato ocorre no problema 1.1 (ver o comando 2 - SQR(4-L*L), onde SQR(4-L*L) ≈ 2).
Também no problema 1.2, ln(x+*h*) ≈ ln(x) provoca o principal erro no cálculo da derivada.

6.3 Overflow / underflow

Este problema ocorre quando o resultado de uma operação provoca um número maior (overflow) ou menor (underflow) que o representável pela palavra do computador.

6.4 Sucessivas multiplicações e divisões

Pode-se 'driblar" o problema de overflow / underflow organizando a ordem das operações.
Ex.: (a.b.c.d)/(x.y.z) pode conduzir a este problema se x.y ou x.y.z ou a.b.c.d forem muito grandes.
Então fazemos, por exemplo, (a/x).(b/y).(c/z).d.

7. Erros de Propagação

Os erros de arredondamentos podem se acumular à medida que uma seqüência de operações matemáticas são efetuadas. Junta-se a isto, outros erros cometidos por interrupção de um processo iterativo infinito ou por tomarmos outra qualquer aproximação.
Teremos uma idéia de como estes erros se propagam.
Consideraremos (no cálculo) x como sendo a soma de X (exato) com o erro Ex.
Analogamente, y = Y + Ey.

7.1. Adição/subtração

0 erro da soma (x±y) será E(x±y) = $\boxed{Ex \pm Ey}$ pois (x±y) = (X+Ex) ± (Y+Ey) = (X±Y) + (Ex±Ey) .
Um exemplo de problema causado por estes arredondamentos é o problema 1.4, onde somamos 10.000 vezes o número 0,0001, que é um binário periódico arredondado pelo computador.

7.2. Multiplicação / divisão

0 erro do produto (x.y) será E(x.y) ≈ $\boxed{X.Ey + Y.Ex}$, pois consideramos Ex.Ey desprezível e
(x.y) = (X+Ex).(Y+Ey) = (X.Y) + X.Ey + YEx + Ex.Ey.
O erro relativo do produto será (X.Ey + YEx) / XY ≈ Ex / X + Ey / Y.

Analogamente, o erro da divisão será: E(x/y) ≈ $\boxed{Ex / Y – X.Ey / Y^2}$.
O erro relativo é dado por Ex / X - Ey / Y.

Anexo 1: **Cálculo de pi**

```
CLS
PRINT "Cálculo de pi: ";
PRINT "2.2.2. ... .2.3.Raiz(2-Raiz(2+Raiz(2+ ... Raiz(2+Raiz(3))...)))."
PRINT " k      n      pi"
n = 6
L = 1
FOR k = 0 TO 35
    p = n * L / 2
    PRINT k; "==> "; n,
    PRINT USING "##.###########"; p
    n = n * 2
    L = SQR(2 - SQR(4 - L * L))
    IF k = 17 THEN Para$ = INPUT$(1):
NEXT k
```

```
Cálculo de pi: 2.2.2. ... .2.3.Raiz(2-Raiz(2+Raiz(2+ ... Raiz(2+Raiz(3))...))).
 k       n       pi
 0 ==>  6       3.00000000000
 1 ==>  12      3.10582852364
 2 ==>  24      3.13262844086
 3 ==>  48      3.13934993744
 4 ==>  96      3.14103198051
 5 ==>  192     3.14145255089
 6 ==>  384     3.14155769348
 7 ==>  768     3.14158391953
 8 ==>  1536    3.14159011841
 9 ==>  3072    3.14159202576
 10 ==>  6144                 3.14159250259
 11 ==>  12288                3.14159250259
 12 ==>  24576                3.14159250259
 13 ==>  49152                3.14159250259
 14 ==>  98304                3.14159250259
 15 ==>  196608               3.14159250259
 16 ==>  393216               3.14159250259
 17 ==>  786432               3.14159250259

 18 ==>  1572864              3.14159250259
 19 ==>  3145728              3.14159250259
 20 ==>  6291456              3.14159250259
 21 ==>  1.258291E+07         3.14159250259
 22 ==>  2.516582E+07         3.14159250259
 23 ==>  5.033165E+07         3.14158678055
 24 ==>  1.006633E+08         3.14158678055
 25 ==>  2.013266E+08         3.14149951935
 26 ==>  4.026532E+08         3.14097476006
 27 ==>  8.053064E+08         3.14027523994
 28 ==>  1.610613E+09         3.13747501373
 29 ==>  3.221225E+09         3.13747501373
 30 ==>  6.442451E+09         3.00000000000
 31 ==>  1.28849E+10          3.00000000000
 32 ==>  2.57698E+10          0.00000000000
 33 ==>  5.153961E+10         0.00000000000
 34 ==>  1.030792E+11         0.00000000000
 35 ==>  2.061584E+11         0.00000000000

 Pressione qualquer tecla para continuar
```

Anexo 2: **Cálculo de pi** (em outros ambientes menos “eficientes”)

	Pentium	(Ver Anexo 1)	Cassio FX	Atari XL
k	$n = 2^k.3$	pi	pi	pi
0	6	3.00000000000	3	3
1	12	3.10582852364	3,105828541	3,10582828
2	24	3.13262844086	3,132628614	3,13262867
3	48	3.13934993744	3,139350204	3,13935045
4	96	3.14103198051	3,141031954	3,14103432
5	192	3.14145255089	2,141452403	3,14145973
6	384	3.14155769348	3,141557544	**3,14159174**
7	768	3.14158391953	3,141584005	3,14176775
8	1536	3.14159011841	**3,14159128**	3,14223706
9	3072	3.14159202576	3,141595035	3,14411358
10	6144	**3.14159250259**	3,1416063	3,14786327
11	12288	3.14159250259	3,141621319	3,19251604
12	24576	3.14159250259	3,141741474	3,2510992
13	49152	3.14159250259	3,142462305	3,47557124
14	98304	3.14159250259	3,143423156	4,9152
15	196608	3.14159250259	3,108645431	9,8304
16	393216	3.14159250259	3,170208743	19,6608
17	786432	3.14159250259	0	39,3216
18	1572864	3.14159250259	0	78,6432
19	3145728	3.14159250259	0	157,2864
20	6291456	3.14159250259	0	314,5728
21	1.258291E+07	3.14159250259	0	629,1456
22	2.516582E+07	3.14159250259		1258,2912
23	5.033165E+07	3.14158678055		2516,5824
24	1.006633E+08	3.14158678055		5033,1648
25	2.013266E+08	3.14149951935		10066,3296
26	4.026532E+08	3.14097476006		20132,6592
27	8.053064E+08	3.14027523994		40265,3184
28	1.610613E+09	3.13747501373		80530,6368
29	3.221225E+09	3.13747501373		161061,273
30	6.442451E+09	3.00000000000		322122,547
31	1.28849E+10	3.00000000000		644245,09
32	2.57698E+10	0.00000000000		1288490,18
33	5.153961E+10	0.00000000000		
34	1.030792E+11	0.00000000000		
35	2.061584E+11	0.00000000000		

Anexos 3 e 4

Anexo 3: **Derivada de *ln*(x)**

```
CLS
INPUT "Derivada de ln(x) para x = "; x
PRINT
FOR i = 1 TO 20
h = 1 / 10 ^ i
d = (LOG(x + h) - LOG(x)) / h
PRINT "Com dx ="; h; "a derivada vale"; d
NEXT i
```

```
Derivada de ln(x) para x = ? 3

Com dx = .1 a derivada vale .3278982
Com dx = .01 a derivada vale .332779
Com dx = .001 a derivada vale .3332778
Com dx = .0001 a derivada vale .3333278
Com dx = .00001 a derivada vale .3333328
Com dx = .000001 a derivada vale .3333333
Com dx = .0000001 a derivada vale .3333333
Com dx = 1E-08 a derivada vale .3333333
Com dx = 1E-09 a derivada vale .3333333
Com dx = 1E-10 a derivada vale .3333333
Com dx = 1E-11 a derivada vale .3333333
Com dx = 1E-12 a derivada vale .3333333
Com dx = 1E-13 a derivada vale .3333325
Com dx = 1E-14 a derivada vale .3333271
Com dx = 1E-15 a derivada vale .3332838
Com dx = 1E-16 a derivada vale .3328501
Com dx = 1E-17 a derivada vale .3252607
Com dx = 1E-18 a derivada vale .3252606
Com dx = 1E-19 a derivada vale 0
Com dx = 1E-20 a derivada vale 0

Pressione qualquer tecla para continuar
```

```
Derivada de ln(x) para x = ? 5

Com dx = .1 a derivada vale .1980263
Com dx = .01 a derivada vale .1998003
Com dx = .001 a derivada vale .19998
Com dx = .0001 a derivada vale .199998
Com dx = .00001 a derivada vale .1999998
Com dx = .000001 a derivada vale .2
Com dx = .0000001 a derivada vale .2
Com dx = 1E-08 a derivada vale .2
Com dx = 1E-09 a derivada vale .2
Com dx = 1E-10 a derivada vale .2
Com dx = 1E-11 a derivada vale .2
Com dx = 1E-12 a derivada vale .2000001
Com dx = 1E-13 a derivada vale .2000006
Com dx = 1E-14 a derivada vale .2000028
Com dx = 1E-15 a derivada vale .2000353
Com dx = 1E-16 a derivada vale .2005774
Com dx = 1E-17 a derivada vale .2059984
Com dx = 1E-18 a derivada vale .2168404
Com dx = 1E-19 a derivada vale 0
Com dx = 1E-20 a derivada vale 0

Pressione qualquer tecla para continuar
```

Anexo 4: **Soma da Série**

```
CLS
PRINT "Soma de 1/(3^n) ==> S1 + S2 + S3 + ... + S20 = ";
S = 0
FOR n = 1 TO 20
  S = S + 1 / (3 ^ n)
NEXT n
PRINT USING "#.##########"; S

PRINT "                ==> S20 + ... + S3 + S2 + S1 = ";
S = 0
FOR n = 20 TO 1 STEP -1
S = S + 1 / (3 ^ n)
NEXT n
PRINT USING "#.##########"; S
```

```
Soma de 1/(3^n) ==> S1 + S2 + S3 + ... + S20 = 0.4999999702
                ==> S20 + ... + S3 + S2 + S1 = 0.5000000000

Pressione qualquer tecla para continuar
```

Anexos 5 e 6

Anexo 5: **10000 parcelas de 0,0001**

```
CLS
PRINT "10000 x 0.0001 = 0.0001 + 0.0001 + ... + 0.0001 =";
S = 0
FOR n = 1 TO 10000
  S = S + .0001
NEXT n
PRINT S
e = S - 1
p = 100 * e
PRINT "Com erro de "; : PRINT USING "#.##########"; e;
PRINT " ou seja "; : PRINT USING "#.########"; p; : PRINT "%"
```

```
10000 x 0.0001 = 0.0001 + 0.0001 + ... + 0.0001 = 1.000054
Com erro de 0.0000535250 ou seja 0.00535250%

Pressione qualquer tecla para continuar
```

Anexo 6: **Raiz quadrada por método iterativo**

```
CLS
   INPUT "     Cálculo  (ITERATIVAMENTE) da Raiz de "; q
   PRINT : PRINT "     ENTER p/ continuar e ESPAÇO p/ parar": PRINT
   x = q / 2
   PRINT "     z ="; q; "/ x       x=(x+z)/2        erro"
10 z = q / x
   x = (x + z) / 2
   PRINT USING "####.#############"; z; x; x * x - q
   IF INPUT$(1) <> " " THEN 10
```

```
     Cálculo  (ITERATIVAMENTE) da Raiz de ? 1.69

     ENTER p/ continuar e ESPAÇO p/ parar

     z = 1.69 / x       x=(x+z)/2        erro
   2.000000000000   1.422500014305   0.333506226540
   1.188049197197   1.305274605751   0.013741739094
   1.294746756554   1.300010681152   0.000027713890
   1.299989342690   1.299999952316  -0.000000181198
   1.300000071526   1.299999952316  -0.000000181198
   1.300000071526   1.299999952316  -0.000000181198

Pressione qualquer tecla para continuar
```