Capítulo 1

Introdução

O objetivo desta disciplina é discutir e aplicar técnicas e métodos numéricos para a resolução de problemas em processos químicos, bioquímicos e indústrias de alimentos. Os métodos apresentados estão presentes em praticamente todas as ferramentas computacionais usadas pelos engenheiros para sintetizar, analisar, controlar e otimizar tais processos. Espera-se que ao final da disciplina o aluno domine estes métodos para fazer bom uso destas ferramentas ou para desenvolver novas ferramentas.

Geralmente os métodos numéricos são implementados nestas ferramentas através de uma linguagem de programação, usualmente FORTRAN, C ou C++. Muitos deles podem ser encontrados em bibliotecas (ou pacotes) numéricos, tais como:

LAPACK (http://www.netlib.org/lapack) – público

BLAS (*Basic Linear Algebra Subprograms*) – público

IMSL (*International Mathematical and Statistical Libraries*) – comercial

NAG (*Numerical Algorithms Group*) – comercial

Mas também estão disponíveis em ambientes dedicados à resolução de problemas genéricos, tais como: MATLAB, SCILAB, MAPLE, MAXIMA e MATHCAD.

1.1 Sistemas numéricos

Como a aritmética em calculadoras e computadores emprega apenas números com uma quantidade finita de dígitos, os cálculos são executados com valores aproximados dos números verdadeiros. Para entender esta aritmética, primeiro trataremos da transformação da base decimal para a binária, usada nos processadores numéricos.

1º Caso: Número Inteiro – N

Algoritmo 1:

a) Identificação da maior potência de 2 contida em N, isto é, determinação de n tal que $2^n \leq N < 2^{n+1}$:

$$n = \text{int}\,[\log_2 (N)]$$

onde int(x) é a parte inteira de x.

b) Determinação dos coeficientes a_i, para $i = 0, 1, ..., n$, tais que: $N = \sum_{i=0}^{n} a_i 2^i$

$P \leftarrow N$

Para $i = 0, 1, ..., n$, faça

$M \leftarrow P/2$

$P \leftarrow \text{int}(M)$

$a_i \leftarrow 2 \cdot (M - P)$

Nota: a operação $2 \cdot [P/2 - \text{int}(P/2)]$ é definida como "P mod 2" ou mod(P,2) ou ainda P % 2 e resulta no resto da divisão inteira de P por 2.

Algoritmo 2:

a) Identificação da maior potência de 2 contida em N, isto é, determinação de n tal que $2^n \leq N < 2^{n+1}$:

$n \leftarrow 0$

$pot \leftarrow 1$

Enquanto pot $\leq N$ faça

$n \leftarrow n + 1$

$pot \leftarrow 2 \cdot pot$

$n \leftarrow n - 1$

b) Determinação dos coeficientes a_i, para $i = 0, 1, ..., n$, tais que: $N = \sum_{i=0}^{n} a_i 2^i$

$pot \leftarrow 2^n$

$a_n \leftarrow 1$

$M \leftarrow N - pot$

Para $i = 1, 2, ..., n$, faça

$pot \leftarrow pot / 2$

se $M < pot$ faça $a_{n-i} \leftarrow 0$

senão faça $a_{n-i} \leftarrow 1$ e $M \leftarrow M - pot$

Exemplo: N = 97 (na base decimal)

Algoritmo 1a: $n = \text{int}\,[\log_2 (97)] = \text{int}[6{,}6] = 6$

Algoritmo 1b:

i	0	1	2	3	4	5	6
M	48,5	24	12	6	3	1,5	0,5
a_i	$a_0 = 1$	$a_1 = 0$	$a_2 = 0$	$a_3 = 0$	$a_4 = 0$	$a_5 = 1$	$a_6 = 1$

Algoritmo 2a:

n	0	1	2	3	4	5	6	7	6
pot	1	2	4	8	16	32	64	128	

Algoritmo 2b:

i	0	1	2	3	4	5	6
pot	64	32	16	8	4	2	1
M	33	1	1	1	1	1	0
a_{n-i}	$a_6 = 1$	$a_5 = 1$	$a_4 = 0$	$a_3 = 0$	$a_2 = 0$	$a_1 = 0$	$a_0 = 1$

Assim: $97\big|_{10} = 1100001\big|_2 = \left(2^6 + 2^5 + 2^0\right)\Big|_{10}$

Exercício: aplicar o Algoritmo 1 para $N = 85$

Algoritmo 1a: $n = \text{int}\,[\log_2 (85)] = \text{int}[6{,}4] = 6$

Algoritmo 1b:

i	0	1	2	3	4	5	6
M	42,5	21	10,5	5	2,5	1	0,5
a_i	$a_0 = 1$	$a_1 = 0$	$a_2 = 1$	$a_3 = 0$	$a_4 = 1$	$a_5 = 0$	$a_6 = 1$

Assim: $85\big|_{10} = 1010101\big|_2 = \left(2^6 + 2^4 + 2^2 + 2^0\right)\Big|_{10}$

Generalizando para qualquer base:

$\underline{n \leftarrow \text{int}[\log_{\text{base}} (N)]}$

$P \leftarrow N$

Para $i = 0, 1, 2, ..., n$, faça

$M \leftarrow P/\text{base}$

$P \leftarrow \text{int}(M)$

$a_i \leftarrow \text{base} \cdot (M - P)$

Nota: a operação base · [*P*/base – int(*P*/base)] é definida como "*P* mod base" ou mod(*P*,base) ou ainda *P* % base e resulta no resto da divisão inteira de *P* por base.

2º Caso: Número fracionário, α, entre 0 e 1

a) Especificar o número de dígitos (N_{dig})

b) Determinar os coeficientes b_i, para $i = 1, 2, ..., N_{dig}$, tais que: $\alpha \cong \sum_{i=1}^{N_{dig}} \frac{b_i}{2^i}$

$M \leftarrow 2 \cdot [\alpha - \text{int}(\alpha)]$

Para $i = 1, 2, ..., N_{dig}$, faça

$P \leftarrow \text{int}(M)$

$M \leftarrow 2 \cdot (M - P)$

$b_i \leftarrow P$

Exemplo: $\alpha = 0{,}8$

a) $N_{dig} = 8$

b)

i	1	2	3	4	5	6	7	8
M	1,6	1,2	0,4	0,8	1,6	1,2	0,4	0,8
$\boldsymbol{b_i}$	$b_1 = 1$	$b_2 = 1$	$b_3 = 0$	$b_4 = 0$	$b_5 = 1$	$b_6 = 1$	$b_7 = 0$	$b_8 = 0$

Assim: $0{,}8|_{10} \cong 0{,}11001100|_2 = \left(\frac{1}{2^1} + \frac{1}{2^2} + \frac{1}{2^5} + \frac{1}{2^6} \right)\Big|_{10} = 0{,}796875|_{10}$

Exercício: $\alpha = 0{,}1$

a) $N_{dig} = 8$

b)

i	1	2	3	4	5	6	7	8
M	0,2	0,4	0,8	1,6	1,2	0,4	0,8	1,6
$\boldsymbol{b_i}$	$b_1 = 0$	$b_2 = 0$	$b_3 = 0$	$b_4 = 1$	$b_5 = 1$	$b_6 = 0$	$b_7 = 0$	$b_8 = 1$

Assim: $0{,}1|_{10} \cong 0{,}00011001|_2 = \left(\frac{1}{2^4} + \frac{1}{2^5} + \frac{1}{2^8} \right)\Big|_{10} = 0{,}0976525|_{10}$

A representação da aritmética de ponto flutuante em computação foi normalizada em 1985 pelo IEEE (*Institute for Electrical and Electronic Engineering*) através da Norma 754-1985.

Definições: *bit* – dígito binário

byte – conjunto de 8 bits

word – ou palavra, é o espaço no computador para armazenar o número. Atualmente na maioria dos computadores:

1 *word* = 32 *bits* = 4 *bytes* ou

1 *word* = 64 *bits* = 8 *bytes*.

Armazenamento de números inteiros:

bit 0: indica o sinal do número, se igual a 0(zero) número positivo e se igual a 1 número negativo;

bits 1 a 31: codificação do número na base binária quando 1 word = 32 bits.

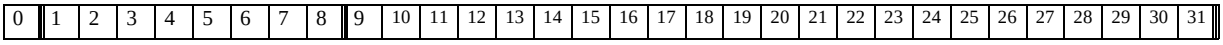
bits 1 a 63: codificação do número na base binária quando 1 word = 64 bits.

Desta forma o maior número inteiro possível é:

$$2^{31} - 1 = 2147483647 \cong 2 \cdot 10^9 \qquad \text{ou} \qquad 2^{63} - 1 \cong 9 \cdot 10^{18}$$

Armazenamento de números reais

a) Precisão simples



bit 0 (*s*): indica o sinal do número, se igual a 0(zero) número positivo e se igual a 1 número negativo;

bits 1 a 8 (*c*): codificação do expoente (ou característica) de 2 do número (igual ao valor representado em binário menos 127, desta forma o maior expoente é 127 = 11111110 – 127 e o menor expoente é –126 = 00000001 – 127. Esta codificação pode ser lida removendo o bit 1 e somando o seu valor ao bit 8 para os expoentes positivos e a representação complementar para os números negativos, com o primeiro bit representando o sinal negativo. A codificação 11111111 é reservada para infinito e 00000000 para indicar que o número não está normalizado, ou seja, que o número antes da vírgula é zero e não 1);

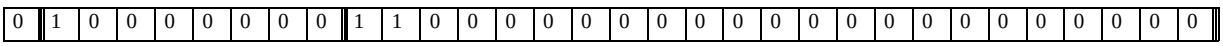
bits 9 a 31 (*f*): codificação da mantissa do número (parte fracionária do número no sistema binário).

$$r = (-1)^s \cdot 2^{c-127} \cdot (1 + f) \ , \text{ quando c} \neq 0$$

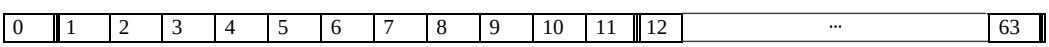
$$r = (-1)^s \cdot 2^{-127} \cdot f \ , \text{ quando c = 0 (forma não-normalizada)}$$

Exemplos:

armazenamento (normalizado) de $(3.5)_{10} = (11.1)_2 = (1.11 \times 2^1)_2$



b) Precisão dupla



bit 0 (*s*): indica o sinal do número, se igual a 0(zero) número positivo e se igual a 1 número negativo

bits 1 a 11 (*c*): codificação do expoente de 2 do número (igual ao valor representado em binário menos 1023, desta forma o maior expoente é 1023 = 11111111110 – 1023 e o menor expoente é –1022 = 00000000001 – 1023);

bits 12 a 63 (*f*): codificação da mantissa do número (parte fracionária do número no sistema binário).

$$r = (-1)^s \cdot 2^{c-1023} \cdot (1+f) \ , \text{ quando c} \neq 0$$

$$r = (-1)^s \cdot 2^{-1023} \cdot f \ , \text{ quando c = 0 (forma não-normalizada).}$$

Como 63 – 12 + 1 = 52 dígitos binários correspondem a algo entre 16 a 17 dígitos decimais (até 2^{-52}), então um número representado neste sistema tem uma precisão de pelo menos 16 dígitos decimais. No caso da precisão simples, este número cai para 7 dígitos decimais (até 2^{-23}).

Exercício: implementar em calculadora ou computador o seguinte algoritmo

$u \leftarrow 1{,}0$

Enquanto $u + 1{,}0 \neq 1{,}0$, faça

    $u \leftarrow u \cdot 0{,}5$

$u \leftarrow u \cdot 2{,}0$

O valor final em u corresponde à precisão da máquina.

1.2 Erros em computação

O menor número positivo que pode ser representado na forma normalizada em precisão dupla é:

$$2^{-1022} \cdot (1+2^{-52}) \cong 10^{-308}$$

Se os cálculos gerarem números de magnitude menor que este valor, teremos um **erro de *underflow*** (além da capacidade mínima da máquina).

O maior número positivo que pode ser representado na forma normalizada em precisão dupla é:

$$2^{1023} \cdot (1+1-2^{-52}) \cong 10^{308}$$

Se os cálculos gerarem números de magnitude maior que este valor, teremos um **erro de *overflow*** (capacidade máxima da máquina excedida).

Erros absolutos e relativos

x: valor exato de um número

x^*: um valor aproximado

Erro absoluto: $\text{EA}_x = |x - x^*|$

Erro relativo: $\text{ER}_x = \left|\frac{x - x^*}{x}\right| = \left|1 - \frac{x^*}{x}\right| = \frac{\text{EA}_x}{|x|}$, para $x \neq 0$

Exemplos:

1) Se $\text{EA}_x = 0{,}1$

a) com $x = 2112{,}9$ tem-se: $x - x^* = \pm 0{,}1 \rightarrow x^* = 2112{,}8$ ou $x^* = 2113$

e $ER_x = 0{,}1/2112{,}9 = 4{,}732832 \cdot 10^{-5} \cong 0{,}005\%$

b) com $x = 5{,}3$ tem-se: $x^* = 5{,}2$ ou $x^* = 5{,}4$

e $ER_x = 0{,}1/5{,}3 = 0{,}018868 \cong 2\%$

2) Efeito da magnitude do número

a) se $x = 0{,}3000 \cdot 10^1$ e $x^* = 0{,}3100 \cdot 10^1$ tem-se $EA_x = 0{,}1$ e $ER_x = 0{,}03333 \cong 3{,}33\%$

b) se $x = 0{,}3000 \cdot 10^{-3}$ e $x^* = 0{,}3100 \cdot 10^{-3}$ tem-se $EA_x = 0{,}1 \cdot 10^{-4}$ e $ER_x = 0{,}03333 \cong 3{,}33\%$

c) se $x = 0{,}3000 \cdot 10^4$ e $x^* = 0{,}3100 \cdot 10^4$ tem-se $EA_x = 0{,}1 \cdot 10^3$ e $ER_x = 0{,}03333 \cong 3{,}33\%$

Ou seja, o erro relativo leva em consideração a magnitude dos valores.

Diz-se que o número x^* se aproxima do valor x com t **algarismos significativos** se t é o maior valor inteiro não negativo para o qual:

$$ER_x < 5 \cdot 10^{-t}$$

Exemplo: com 4 algarismos significativos

$x = 0{,}1 \rightarrow ER_x \cdot |x| = |x - x^*| = EA_x < 5 \cdot 10^{-5}$

$x = 100 \rightarrow ER_x \cdot |x| = |x - x^*| = EA_x < 0{,}05$

$x = 10000 \rightarrow ER_x \cdot |x| = |x - x^*| = EA_x < 5$

Representando um número x em aritmética de ponto flutuante com t dígitos na base 10:

$$x = f_x \cdot 10^e + g_x \cdot 10^{e-t} \quad \text{com } 0{,}1 \le f_x < 1 \text{ e } 0 \le g_x < 1$$

Exemplo: $\boldsymbol{t} = 4$ e $x = 234{,}57$, logo $x = 0{,}2345 \cdot 10^3 + 0.7 \cdot 10^{-1} \rightarrow e = 3$, $f_x = 0{,}2345$ e $g_x = 0{,}7$

Se o número for simplesmente truncado**,** o valor armazenado de x será: $x^* = f_x \cdot 10^e$, apresentando:

Erro de Truncamento Absoluto: $EA_x = |g_x| \cdot 10^{e-t} \le 10^{e-t}$

Erro de Truncamento Relativo: $ER_x = \left| \dfrac{g_x \cdot 10^{e-t}}{f_x \cdot 10^e + g_x \cdot 10^{e-t}} \right| < \dfrac{10^{e-t}}{f_x \cdot 10^e} < \dfrac{10^{e-t}}{0{,}1 \cdot 10^e} = 10^{1-t}$.

Se o número for arredondado, o valor armazenado de x será:

$x^* = f_x \cdot 10^e + \begin{cases} 0 & \text{se } g_x < 0.5 \\ 10^{e-t} & \text{se } g_x \ge 0.5 \end{cases}$, apresentando ***Erros de Arredondamento Absoluto*** **e Relativo**:

i) se $g_x < 0{,}5$: $EA_x = |g_x| \cdot 10^{e-t} \le 0{,}5 \cdot 10^{e-t}$

$$\text{e } \mathrm{ER}_x = \left| \frac{g_x \cdot 10^{e-t}}{f_x \cdot 10^{e} + g_x \cdot 10^{e-t}} \right| < \frac{0{,}5 \cdot 10^{e-t}}{|\, f_x \cdot 10^{e} \,|} < \frac{0{,}5 \cdot 10^{e-t}}{0{,}1 \cdot 10^{e}} = 0{,}5 \cdot 10^{1-t} = 5 \cdot 10^{-t}$$

ii) se $g_x \geq 0{,}5$: $\mathrm{EA}_x = |\, g_x \cdot 10^{e-t} - 10^{e-t} \,| \leq |\, 0{,}5 \cdot 10^{e-t} - 10^{e-t} \,| \leq 0{,}5 \cdot 10^{e-t}$

$$\text{e } \mathrm{ER}_x = \left| \frac{(g_x - 1) \cdot 10^{e-t}}{f_x \cdot 10^{e} + g_x \cdot 10^{e-t}} \right| < \frac{0{,}5 \cdot 10^{e-t}}{|\, f_x \cdot 10^{e} \,|} < \frac{0{,}5 \cdot 10^{e-t}}{0{,}1 \cdot 10^{e}} = 0{,}5 \cdot 10^{1-t} = 5 \cdot 10^{-t}$$

Resumindo, se: $x = f_x \cdot 10^{e} + g_x \cdot 10^{e-t}$ com $0{,}1 \leq f_x < 1$ e $0 \leq g_x < 1$, em que t é o número de dígitos, então:

Erros de Truncamento: $\begin{cases} \text{Absoluto: } \mathrm{EA}_x < 10^{e-t} \\ \text{Relativo: } \mathrm{ER}_x < 10^{1-t} \end{cases}$

Erros de Arredondamento: $\begin{cases} \text{Absoluto: } \mathrm{EA}_x < \dfrac{10^{e-t}}{2} \\ \text{Relativo: } \mathrm{ER}_x < \dfrac{10^{1-t}}{2} \end{cases}$

Erros nas operações algébricas fundamentais

Sejam x e y dois números reais positivos que apresentam erros absolutos máximos a e b, respectivamente. Então os erros numéricos relativos em cada um destes números são:

para x: $p = \dfrac{a}{x}$

e para y: $q = \dfrac{b}{y}$.

As seguintes operações aplicadas a esses números são definidas:

i) Soma – o maior valor que a soma $(x + y)$ pode assumir é: $(x + y) + (a + b)$ e o menor valor é: $(x + y) - (a + b)$, assim:

$$(x + y) - (a + b) \leq (x^* + y^*) \leq (x + y) + (a + b).$$

Desse modo: $\mathrm{EA}_{x+y} \leq (a + b)$ e $\mathrm{ER}_{x+y} \leq \dfrac{(a+b)}{|x+y|}$

ii) Subtração – o maior valor que a subtração $(x - y)$ pode assumir é: $(x - y) + (a + b)$ e o menor valor é: $(x - y) - (a + b)$, assim:

$$(x - y) - (a + b) \leq (x^* - y^*) \leq (x - y) + (a + b).$$

Desse modo: $\mathrm{EA}_{x-y} \leq (a + b)$ e $\mathrm{ER}_{x-y} \leq \dfrac{(a+b)}{|x-y|}$.

iii) Produto – o maior valor $(x \cdot y)$ pode assumir é:

$$(x+a)\cdot(y+b) = (x+p\cdot x)\cdot(y+q\cdot y) = x\cdot y\cdot(1+p)\cdot(1+q) = x\cdot y\cdot(1+p+q+p\cdot q)$$

e o menor valor é: $(x-a)\cdot(y-b) = x\cdot y\cdot(1-p)\cdot(1-q) = x\cdot y\cdot(1-p-q+p\cdot q)$

Assim: $x\cdot y\cdot(1-p-q+p\cdot q) \le x^*\cdot y^* \le x\cdot y\cdot(1+p+q+p\cdot q)$, ou seja:

$x\cdot y\cdot(p\cdot q-p-q) \le x^*\cdot y^* - x\cdot y \le x\cdot y\cdot(p\cdot q+p+q)$. Permitindo identificar que:

$$\text{EA}_{x\cdot y} \le |x\cdot y|\cdot(p\cdot q+p+q) \cong |x\cdot y|\cdot(p+q) \text{ e } \text{ER}_{x\cdot y} \le (p\cdot q+p+q) \cong p+q$$

iv) <u>Divisão</u> – o maior valor (x/y) pode assumir é:

$\frac{x+a}{y-b} = \frac{x+p\cdot x}{y-q\cdot y} = \frac{x}{y}\cdot\left(\frac{1+p}{1-q}\right)\cdot\left(\frac{1+q}{1+q}\right) = \frac{x}{y}\cdot\left(\frac{1+p+q+p\cdot q}{1-q^2}\right)$ e o menor valor é:

$\frac{x-a}{y+b} = \frac{x-p\cdot x}{y+q\cdot y} = \frac{x}{y}\cdot\left(\frac{1-p}{1+q}\right)\cdot\left(\frac{1-q}{1-q}\right) = \frac{x}{y}\cdot\left(\frac{1-p-q+p\cdot q}{1-q^2}\right)$.

Assim: $\frac{x}{y}\cdot\left(\frac{1-p-q+p\cdot q}{1-q^2}\right) \le \frac{x^*}{y^*} \le \frac{x}{y}\cdot\left(\frac{1+p+q+p\cdot q}{1-q^2}\right)$, ou seja:

$\frac{x}{y}\cdot\left(\frac{1-p-q+p\cdot q}{1-q^2}-1\right) \le \frac{x^*}{y^*}-\frac{x}{y} \le \frac{x}{y}\cdot\left(\frac{1+p+q+p\cdot q}{1-q^2}-1\right)$, rearranjando:

$\frac{x}{y}\cdot\left(\frac{q^2-p-q+p\cdot q}{1-q^2}\right) \le \frac{x^*}{y^*}-\frac{x}{y} \le \frac{x}{y}\cdot\left(\frac{q^2+p+q+p\cdot q}{1-q^2}\right)$, mas:

$q^2-p-q+p\cdot q = (p+q)\cdot(q-1)$ e $q^2+p+q+p\cdot q = (p+q)\cdot(q+1)$, logo:

$-\frac{x}{y}\cdot\left(\frac{p+q}{1+q}\right) \le \frac{x^*}{y^*}-\frac{x}{y} \le \frac{x}{y}\cdot\left(\frac{p+q}{1-q}\right)$. Permitindo identificar que:

$$\text{EA}_{x/y} \le \left|\frac{x}{y}\right|\cdot\left(\frac{p+q}{1-q}\right) \cong \left|\frac{x}{y}\right|\cdot(p+q) \text{ e } \text{ER}_{x/y} \le \frac{p+q}{1-q} \cong p+q \ .$$

Lista de exercícios

1. Transforme os números reais abaixo (todos expressos na base decimal) para a base binária, adotando em todos os exemplos 8 dígitos após a vírgula: (a) 168,995889; (b) 0,34135; (c) 0,021922.
2. Transforme os números binários abaixo para a base decimal:

   (a) 101101; (b) 110101011; (c) 0,1100011; (d) 0,11111111.
3. Ache um número na base 2 que aproxime o número π com o menor número de dígitos apresentando um erro absoluto não superior a 10^{-3}. Refaça o exemplo para uma aproximação que apresente um erro relativo inferior a 0,01%. Compare e discuta os dois resultados encontrados.
4. Considere que se tem um aparato digital que armazena os números em aritmética de ponto flutuante com quatro dígitos em base decimal. O acumulador (onde são executadas as operações) apresenta precisão dupla (8 dígitos portanto!) e simplesmente trunca os

números acumulados. Dados os números: $x = 0{,}4523 \cdot 10^{4}$; $y = 0{,}2115 \cdot 10^{-3}$ e $z = 0{,}2583 \cdot 10^{1}$, verifique os resultados das seguintes operações executadas neste aparato e apresente, em cada caso, o erro absoluto e o erro relativo resultante:

(a) $x + y + z$; (b) $\frac{x}{z}$; (c) $x - y$; (d) $x - y - z$; (e) $\frac{(x \cdot y)}{z}$; (f) $\left(\frac{x}{z}\right) \cdot y$; (g) $\left(\frac{y}{z}\right) \cdot x$.

Compare e discuta os resultados dos itens (e); (f) e (g).

5. Os números $x = 2$ e $y = 1{,}76$ apresentam, respectivamente, os erros absolutos $a = 0{,}5$ e $b = 0{,}1$. Sorteie um valor de x^{*} e um valor de y^{*} que estejam contidos entre os valores mínimos e máximos de x e y, respectivamente. Calcule os valores máximos dos erros absolutos e relativos das operações abaixo listadas e verifique se os erros destas mesmas operações aplicadas aos valores sorteados de x e y estão contidos dentro destes limites.

(a) $x + y$; (b) $x - y$; (c) $x \cdot y$; (d) $\frac{x}{y}$; (e) x^{k} com $k > 0$; (f) x^{k} com $k < 0$;

(g) $\ln(x)$; (h) e^{x} ; (i) $\cos(x)$; (j) $\text{sen}(x)$; (k) $\text{tg}(x - 1)$.

6. Refaça o Exercício 5 sabendo-se que no lugar dos erros absolutos são conhecidos os erros relativos de x e y iguais, respectivamente, a 20% e 15%.