IPS Instituto Politécnico de Setúbal
Escola Superior de Tecnologia de Setúbal

César Rodrigo Fernández

Sebenta de Análise Numérica

Para alunos da Licenciatura em Engenharia Informática

16 de Março de 2020

César Rodrigo Fernández
ESTS, Instituto Politécnico de Setúbal
Departamento de Matemática

1

Representações Numéricas e Erros

A forma como fazemos cálculos está intimamente relacionada com a forma como representamos os números. O estudo da tabuada e dos algoritmos de soma, multiplicação e divisão ensinados na escola estão justificados pela nossa representação posicional dos números em base 10. Se mantivéssemos o sistema de representação numérica romano, a aritmética (soma, produto, divisão, etc) seria mais complicada de executar do que com a representação posicional na atual notação (com os dez numerais arábigos). Basta com pensar em como poderíamos somar CMXLIX com DXXVIII sem antes fazer uma transformação à notação decimal. Isto ilustra que a forma em que os números são representados condiciona o seu uso posterior. Da mesma forma, para uma boa utilização das ferramentas computacionais é preciso perceber como os números são representados internamente numa máquina de computação científica e as implicações destas representações quando se implementam algoritmos numéricos.

1.1 Representações polinomiais dos números naturais e reais.

Para representar números, na escrita, usamos um sistema posicional com base 10: Há dez símbolos diferentes $(0, 1, \ldots, 9)$, os numerais, e o seu significado depende da posição em que estes símbolos aparecem dentro da escrita. Se um símbolo está situado k passos à esquerda da posição básica (em geral marcada pela presença do ponto decimal), representa um valor 10^k vezes maior do que na posição básica, ou 10^k vezes menor se situado à direita. Assim a sequência de algarismos 5312.45 representa o número:

$$5 \cdot 10^3 + 3 \cdot 10^2 + 1 \cdot 10^1 + 2 \cdot 10^0 + 4 \cdot 10^{-1} + 5 \cdot 10^{-2}$$

Se se evitam os zeros à esquerda antes da posição básica e os zeros à direita após o ponto decimal, e se inclui um símbolo (o traço) para indicar a mudança de sinal do número, temos um sistema de numeração baseado num conjunto pequeno de símbolos (os numerais, o ponto decimal, e o traço) onde não há duas representações diferentes para o mesmo número (com exceção do zero, representado como 0. e como $-0.$), e onde, feita uma escolha dum nível de precisão (erro tolerável), qualquer número real pode ser representado através duma coleção finita de algarismos, com um erro menor que o nível de precisão escolhido.

Há uma variante deste sistema de representação numérica que é comum e podemos observar nas calculadoras de mão: a notação científica. Esta versão situa o ponto decimal sempre após o primeiro algarismo não nulo, e multiplica a expressão resultante com um termo da forma 10^k, sendo k um inteiro. Assim:

$$3.5689\,E\,2 \to 3.5689 \cdot 10^2 = 356.89, \quad -1.213\,E - 3 \to -1.213 \cdot 10^{-3} = -0.001213$$

(Não se deve confundir o E "expoente" nesta expressão com a constante de Euler e). Como veremos mais à frente, uma característica importante desta notação é o conceito de algarismos significativos.

A escolha do dez é provavelmente devida ao facto de termos esse número de dedos nas mãos. Se a matemática fosse uma tarefa realizada só com dedos, é uma escolha lógica. No entanto é um sistema de numeração aprendido, não um sistema associado de forma necessária à máquina "homem", se admitimos que este faz matemática através das suas capacidades racionais, e não através dos dedos. Outras escolhas são possíveis. Nos computadores com memória binária, a estrutura básica de informação é o "bit", que tem dois possíveis estados, ligado ou desligado. Por isto as escolhas mais adequadas nestas máquinas são as representações de números em base 2,4,8 ou outras potências de 2.

Representação escrita de números naturais

Nos números naturais (inteiros positivos) são conhecidas desde antigo uma série de propriedades e algoritmos. Um dos mais destacados é o algoritmo de divisão com resto. Dados um par de números naturais $n \geq 1$, $b > 1$, sabemos que a divisão $n \div b$ produz um quociente $c < n$ e um resto $0 \leq a < b$. Nomeadamente, temos, de maneira única:

$$n = a + b \cdot c, \qquad 0 \leq a < b, \quad c < n \text{ inteiros}$$

Como o quociente é menor que o número originalmente dado, se repetimos a divisão entre b agora para o novo quociente, depois de repetir o processo um número finito de vezes (k vezes) chegamos ao primeiro instante em que o quociente resultante c_k seja nulo, sendo então o resto a_k não nulo (porque $0 \neq c_{k-1} = a_k + b \cdot c_k$). Chegamos assim a uma expressão:

$$\begin{aligned} n &= a_0 + b \cdot c_0 = a_0 + b \cdot (a_1 + b \cdot c_1) = a_0 + b \cdot (a_1 + b \cdot (a_2 + b \cdot c_2)) = \ldots = \\ &= a_0 + b \cdot (a_1 + b \cdot (a_2 + b \cdot (a_3 + b \cdot (a_4 + \ldots)))) \end{aligned}$$

Ou escrito de outra maneira:

$$n = a_0 + a_1 \cdot b + a_2 \cdot b^2 + a_3 \cdot b^3 + \ldots + a_k \cdot b^k, \qquad a_k \neq 0$$

Assim qualquer número natural $n \geq 1$ determina de maneira única uma sequência de valores $(a_k, \ldots, a_1, a_0)$, todos eles inteiros $0 \leq a_i < b$, os chamados **algarismos do número n em base b**, sendo o primeiro deles a_k não nulo e chamado o **algarismo mais significativo**. Diremos que o natural n **tem expoente k em base b**.

Para poder representar a sequência de algarismos basta agora escolher um conjunto de b símbolos diferentes, cada um deles a representar cada um dos inteiros $0 \leq a < b$. No caso da base $b = 10$, o conjunto de símbolos será o dos **numerais arábigos** $\mathcal{N}_{10} = \{0, 1, 2, \ldots, 9\}$, conjunto de símbolos fixados no século XVI na Europa, como evolução de símbolos numerais arábigos mais antigos, e procedentes da matemática indiana.

Definição 1.1 (Sistema de numeração posicional). *Seja b um número natural maior que 1 (a **base**) e $\mathcal{N}_b$ um conjunto de b símbolos (os **numerais**), onde cada um está associado a cada um dos valores naturais $\{0, 1, 2, \ldots, b-1\} \subset \mathbb{N}$ menores que b (os algarismos). Diremos que a sequência de numerais $a_k \ldots a_2 a_1 a_0$ (onde $a_k \neq 0$) é a representação escrita em notação posicional em base b do número*

$$\boxed{n = a_0 + b \cdot (a_1 + b \cdot (a_2 + \ldots)) = a_0 + a_1 \cdot b + a_2 \cdot b^2 + \ldots + a_k \cdot b^k}$$

(aqui a soma e produto não simboliza uma operação nos numerais, símbolos gráficos, senão uma operação nos números por eles representados)

Para evitar confusão sobre a base em que trabalhamos, na escrita usaremos os numerais correspondentes a cada algarismo e representaremos o número n como uma sequência de numerais, entre parêntesis e com indicação específica da base usada, segundo o modelo:

$$(a_k \dots a_2 a_1 a_0)_b$$

Só se a base está claramente indicada (por exemplo quando assumimos que estamos a usar a notação decimal) deixaremos de lado os parêntesis e a indicação da base.

Segundo sabemos pela unicidade do quociente e resto ao dividir entre b, este sistema de representação numérica permitirá representar qualquer número natural $n \geq 1$ de maneira única.

Nota 1.2. Não devemos confundir número natural (valor que serve para contar elementos dum conjunto) com algarismo (cada um dos números entre 0 e $b-1$ obtidos na decomposição dum natural) ou com numeral (símbolos gráficos usados para representar algarismos na escrita).

Nota 1.3. A expressão algébrica $a_0+b\cdot(a_1+b\cdot(a_2+b\cdot(a_3+b\cdot(a_4+\dots))))$ é chamada **esquema de Hörner** para o cálculo duma expressão polinomial. Exige só k somas e k produtos, em comparação com a notação clássica $a_0 + a_1 \cdot b + a_2 \cdot b^2 + \dots + a_k \cdot b^k$, que exige k somas e $1 + 2 + 3 + \dots + k = k(k+1)/2$ produtos.

Os sistemas de representação de inteiros mais usados são os que utilizam a base $b = 10$, e a base $b = 2$. Para bases $b < 10$ resulta habitual usar como numerais uma parte dos símbolos arábigos, aqueles que representam os valores entre 0 e $b-1$. Assim um número escrito na notação binária (base 2) está representado na escrita por uma sequência de símbolos $0, 1$ onde o algarismo mais significativo (símbolo inicial) é o 1:

$$(1101)_2 = 1 + 2 \cdot (0 + 2 \cdot (1 + 2 \cdot 1)) = 13$$

Nesta notação o número **não deve ser lido como dízima**, assim não diremos "o número mil e cento e um em binário é treze", senão "o número um um zero um em binário é treze".

Um método habitual para introduzir valores no computador através da notação binária (por exemplo na linguagem C, Octave ou algumas edições de Matlab) seria `0b1101`, representando assim o valor 13.

Nota 1.4. Nos casos em que $b > 10$, é necessário introduzir novos símbolos (numerais) para representar os algarismos $0, 1, \dots, b-1$. Por exemplo, em base 16 (**notação hexadecimal**) o número $(28)_{10} = 2 \cdot 10^1 + 8 \cdot 10^0 = 1 \cdot 16^1 + 12 \cdot 16^0$ é representado como

$$(1C)_{16}$$

onde usamos os símbolos (numerais hexadecimais) $\mathcal{N}_{16} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A(=10), B(=11), C(=12), D(=13), E(=14), F(=15)\}$

Se tivéssemos escolhido como símbolos $\mathcal{N}_{16} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15\}$ ficaríamos obrigados a escrever $((1)(12))_{16}$, onde os parêntesis seriam precisos para distinguir $((1)(1)(2))_{16} = 1 \cdot 16^2 + 1 \cdot 16^1 + 2 \cdot 16^0$ de $((11)(2))_{16} = 11 \cdot 16^1 + 2 \cdot 16^0$ e de $((1)(12))_{16} = 1 \cdot 16^1 + 12 \cdot 16^0$.

Muitas linguagens de programação admitem também a introdução de números com notação hexadecimal, por exemplo em C ou Octave ou Matlab está permitida a introdução na consola de $0xA12$ para indicar $(A12)_{16} = 2 + 16 \cdot (1 + 16 \cdot 10) = 2578$

Nos casos em que a base é $100 = 10^2$, resulta natural utilizar como numerais $\mathcal{N}_{100}$ os dígrafos $\{00, 01, 02, 03, 04, 05, 06, \dots, 99\}$. Uma vantagem é que transformar um número em base 100 a base 10 é simplesmente transformar cada dígrafo no par de numerais decimais correspondentes:

$$((10)(54)(61))_{100} = 10 \cdot (10^2)^2 + 54 \cdot (10^2)^1 + 61 = (105461)_{10}$$

O mesmo é válido para 10^3 (se usamos os trígrafos ou ternas $000, 001, \dots, 999$ como alfabeto), ou para bases $4 = 2^2$, $8 = 2^3$, etc.

De maneira similar ao paralelismo da representação escrita entre as bases 10 e 100, existe uma relação evidente entre a representação binária e a representação octal com trígrafos (estude qual é esta relação e trate de justificá-la). Por exemplo, se queremos exprimir um número em base 8 (notação octal), podemos usar $\{0,1,2,3,4,5,6,7\}$ como alfabeto, mas também as ternas $\{000,001,010,011,100,101,110,111\}$. Assim, transformar a binário é simples:

$$(5632)_8 = ((101)(110)(011)(010))_8 = (101110011010)_2$$

onde (101) é uma terna que representa $1\cdot 2^2+0\cdot 2^1+1=5$, (110) representa $1\cdot 2^2+1\cdot 2^1+0=6$, etc.

Na realidade, num certo sentido, mais do que com bits, podemos pensar que os computadores trabalham com unidades de informação que admitem 2^{32} ou 2^{64} estados possíveis (depende do tamanho da "memory word", grupo de bits que são processados conjuntamente na CPU e que indica se a máquina é de 32 ou 64 bits). Poder-se-ia pensar que a base de trabalho devia ser então $b=2^{32}$ ou $b=2^{64}$. A todos os efeitos, não faz diferença com o formato binário, dado que os algarismos nestas bases podem-se representar como sequências de 32 ou 64 zeros/uns, em forma análoga ao que fizemos em base $2^3=8$.

Exemplo

Se queremos representar os números entre 1 e 10 em base $b=2$ e usamos como numerais $\mathcal{N}_2=\{0,1\}$ temos o seguinte resultado:

Notação decimal	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
Notação binária	*1*	*10*	*11*	*100*	*101*	*110*	*111*	*1000*	*1001*	*1010*

Se queremos representar os mesmos números em base 8 podemos escolher como numerais os símbolos $\mathcal{N}_8=\{0,1,2,3,4,5,6,7\}$ ou, tendo em conta a tabela anterior, também podemos escolher como numerais os símbolos (trígrafos) $\mathcal{N}_{b8}=\{000,001,010,011,100,101,110,111\}$

Notação decimal	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
Notação octal	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*10*	*11*	*12*
Notação trígrafos	*001*	*010*	*011*	*100*	*101*	*110*	*111*	*001 000*	*001 001*	*001 010*

A notação octal foi obtida a dividir o número entre 8 e retirar o quociente e resto. A sequência de restos produz os algarismos octais associados ao número n, mas ordenados desde o menos significativo ao mais significativo. Na representação escrita em base octal devemos usar a sequência de numerais correspondentes a estes algarismos, escritos em ordem oposta.

Se queremos representar o número $n=2020$ em base $b=8$, começamos por determinar os algarismos associados. Para isto iteramos várias vezes a divisão com resto para determinar os quocientes c_i e restos a_i

(÷8)	$252+\frac{4}{8}$	$31+\frac{4}{8}$	$3+\frac{7}{8}$	$0+\frac{3}{8}$
$n=2020$ c_i	*252*	*31*	*3*	*0*
a_i	*4*	*4*	*7*	*3*

Em notação octal se usamos numerais $\mathcal{N}_8$, o número n seria representado por $(3744)_8$. Se usamos numerais $\mathcal{N}_{b8}$, o mesmo número n seria representado por $(011\,111\,100\,100)$

O mesmo número, em notação binária, seria representado por $(11111100100)_2$, basta separar cada trígrafo em 3 numerais binários, sendo desnecessário iterar a divisão entre 2 onze vezes.

Se pretendemos dar agora o mesmo número em notação hexadecimal, poderíamos usar divisão com resto entre $b=16$ ou aproveitar a representação conhecida em notação binária. Se fixamos como numerais os símbolos ("tetragrafos")

$$\mathcal{N}_{b16}=\{0000,0001,0010,0011,0100,0101,0110,0111,1000,1001,1010,1011,1100,1101,1110,1111\}$$

temos a seguinte representação hexadecimal para $n=2020$:

$$(0111\,1110\,0100)_{16}$$

ou se usamos como numerais os símbolos $\mathcal{N}_{16} = \{0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F\}$ *temos a seguinte representação hexadecimal:*

$$(7E4)_{16}$$

Operações em base b.

Em notação posicional em base b a soma, a multiplicação e a divisão podem efetuar-se através de algoritmos análogos aos aprendidos na escola para base 10. Assim, por exemplo, multiplicar por $b = (10)_b$ consiste em trasladar o ponto decimal uma posição à direita, e a multiplicação faz-se através de tabuadas de soma e multiplicação que podem ser decoradas.

Exemplo

Vamos fazer o produto dos números $(1001101)_2$ *e* $(110)_2$ *em representação binária*

$$\begin{array}{r} 1\ 0\ 0\,1\,1\,0\,1 \\ \times_2 \quad 1\,1\,0 \\ \hline 0\ 0\ 0\,0\,0\,0\,0 \\ 1\,0\ 0\ 1\,1\,0\,1 \\ +_2\ 1\,0\,0\ 1\ 1\,0\,1 \\ \hline 1\,1\,1\ 0\ 0\,1\,1\,1\,0 \end{array}$$

$$\begin{cases} \textit{Tabuada da soma:} \\ 0+0=0,\ 0+1=1 \\ 1+0=1,\ 1+1=(10)_2 \end{cases} \qquad \begin{cases} \textit{Tabuada do produto:} \\ 0\times 0=0,\ 0\times 1=0 \\ 1\times 0=0,\ 1\times 1=1 \end{cases}$$

onde na soma, há que pensar "$1+1=(10)_2$*, ponho o* 0*, e levo* 1*".*

O método acima indicado é formalmente igual ao habitual para soma de números dados em base 10. O resultado diz-nos que o produto de $(1001101)_2 = 1+0\cdot 2+1\cdot 4+1\cdot 8+0\cdot 16+0\cdot 32+1\cdot 64 = 1+4+8+64 = 77$ *com* $(110)_2 = 0+1\cdot 2+1\cdot 4 = 6$ *é* $(111001110)_2 = 1\cdot 2+1\cdot 4+1\cdot 8+1\cdot 64+1\cdot 128+1\cdot 256 = 2+4+8+64+128+256 = 462$*. Para a divisão o método standard também serve.*

Exemplo

Vamos fazer o produto dos números $(2311)_4$ *e* $(213)_4$ *em representação de base 4:*

$$\begin{array}{r} 2\,3\,1\,1 \\ \times \quad 2\,1\,3 \\ \hline 2\,0\,1\,3\,3 \\ 2\,3\,1\,1 \\ +\ 1\ 1\ 2\,2\,2 \\ \hline 1\ 2\ 3\ 2\,1\,0\,3 \end{array}$$

$$\begin{cases} \textit{Tabuada da soma:} \\ 0+0=0 & 0+1=1 & 0+2=2 & 0+3=3 \\ 1+0=1 & 1+1=2 & 1+2=3 & 1+3=(10)_4 \\ 2+0=2 & 2+1=3 & 2+2=(10)_4 & 2+3=(11)_4 \\ 3+0=3 & 3+1=(10)_4 & 3+2=(11)_4 & 3+3=(12)_4 \end{cases}$$

$$\begin{cases} \textit{Tabuada do produto:} \\ 0\times 0=0 & 0\times 1=0 & 0\times 2=0 & 0\times 3=0 \\ 1\times 0=0 & 1\times 1=1 & 1\times 2=2 & 1\times 3=3 \\ 2\times 0=0 & 2\times 1=2 & 2\times 2=(10)_4 & 2\times 3=(12)_4 \\ 3\times 0=0 & 3\times 1=3 & 3\times 2=(12)_4 & 3\times 3=(21)_4 \end{cases}$$

Uma simples comprovação em notação de base 10, mostra que estamos a afirmar que o produto de $1{+}4(1{+}4(3{+}4{\cdot}2)) = 181$ *com* $3{+}4(1{+}4{\cdot}3) = 39$ *é* $3{+}4(0{+}4(1{+}4(2{+}4(3{+}4(2{+}4{\cdot}1))))) = 7059$*, o qual é o resultado correto.*

Não vamos insistir nos algoritmos básicos de soma, produto, diferença ou divisão com resto. No entanto, se conhecemos a tabuada da soma e do produto para todos os números $\{0,1,\ldots,b-1\}$, somos capazes de fazer estas operações em qualquer base sem termos que passar a uma representação decimal.

Representação de valores negativos

O quê podemos fazer se queremos representar o zero? O zero pode ser representado simplesmente pelo símbolo numeral que fixamos para este algarismo. Com isto rompemos o princípio de que nas nossas representações devíamos começar sempre com um algarismo não nulo.

E se queremos representar também inteiros negativos?

A opção que parece mais natural é aproveitar a simetria: Para cada inteiro negativo k existe o correspondente oposto. Inserimos então na nossa notação dois símbolos $\{+,-\}$. O primeiro indica que estamos a escrever um número positivo. O segundo que estamos a escrever um número negativo. Esta é a notação habitualmente usada. Se aceitamos a possibilidade de escrever o zero, o método de inserir um sinal leva à existência dum "zero positivo" e um "zero negativo", duas maneiras diferentes de escrever o mesmo número zero.

Num sistema de numeração onde conseguimos representar só um conjunto finito de números diferentes, uma boa ideia é tratar usar metade das opções para representar valores positivos, e a outra metade para representar os correspondentes negativos (os opostos). Podemos fazé-lo com b numerais através do método de representação com excesso:

Definição 1.5. *Chamamos representação de inteiros $-N \leq k < b^p - N$ em base b através de p numerais com* ***excesso em*** *N a representação do número pelos p algarismos em base b $(a_{p-1} \dots a_1 a_0)$ associados a $k+N$.*

Tendo em conta que existem b^p opções, uma ideia frequente é usar o excesso em $\frac{1}{2}b^p$. Quando b é par, isto supõe simplesmente somar $b/2$ unidades no algarismo da casa p.

Uma vantagem é que a ordem dos números corresponde com a ordem lexicográfica. Uma desvantagem é que a soma e produto não resultam tão simples.

Exemplo

O número 53 *tem representação octal* $(65)_8$. *Se usamos 3 numerais octais, este número é representado com excesso em* $\frac{1}{2}8^3 = (400)_8$ *como* (465).

Para o número -53, *a representação com excesso em* $\frac{1}{2}8^3$ *exige calcular* $\frac{1}{2}8^3 - 53$, *que escrito em octal seria* $(400)_8 - (065)_8 = (313)_8$. *O número* -53 *representado com excesso em* $\frac{1}{2}8^3$ *é escrito por* (313).

Representação escrita de números fracionários

Números naturais são os inteiros positivos $\mathbb{N} = \{1, 2, 3, \dots\}$. Sabemos que resulta possível representá-los de forma única, como sequência finita de numerais (o primeiro deles não nulo), através dos correspondentes algarismos em base b.

Chamaremos **números fracionários** todos os valores reais situados entre 0 e 1. Podemos denotar este conjunto como $\mathbb{F}$

$$\mathbb{F} = \{x \in \mathbb{R} : 0 < x < 1\} =]0, 1[$$

Qualquer número real positivo é um número natural, um número fracionário, ou uma soma de natural e fracionário. Se incorporamos o zero ao mesmo tempo como natural e como fracionário, podemos pensar que os naturais são aqueles reais positivos com parte fracionária nula, e que os números fracionários são aqueles reais positivos com parte inteira nula. Temos assim uma decomposição (única) para os reais positivos:

$$x = n + f, \qquad n = \lfloor x \rfloor \in \mathbb{N} \cup \{0\}, \quad f = \mathrm{fr}(x) \in \mathbb{F} \cup \{0\} = [0, 1[$$

Os símbolos $\lfloor \cdot \rfloor$ e $\mathrm{fr}(\cdot)$ são usados para indicar a determinação da parte inteira e da fração associadas a qualquer número positivo x.

Ao multiplicar um número fracionário $f \in]0, 1[$ com um natural $b > 1$ temos $b \cdot f \in]0, b[$. A parte inteira $c_1 = \lfloor f \cdot b \rfloor \in \{0, 1, \dots, b-1\}$ é chamada o primeiro algarismo do número f em base b. Ao considerarmos a fração restante $f_1 = f \cdot b - \lfloor f \cdot b \rfloor$ obtemos

$$f \cdot b = \lfloor f \cdot b \rfloor + f_1 = c_1 + f_1 \to f = \frac{1}{b}(c_1 + f_1)$$

podemos voltar a multiplicar a nova fração f_1 com b para obter uma nova parte inteira c_2 e fração f_2. Temos:

$$\boxed{f = \frac{1}{b}(c_1 + f_1) = \frac{1}{b}(c_1 + \frac{1}{b}(c_2 + f_2)) = \frac{1}{b}(c_1 + \frac{1}{b}(c_2 + \frac{1}{b}(c_3 + f_3))) = \dots}$$

Portanto:

$$f = c_1 \cdot b^{-1} + c_2 \cdot b^{-2} + c_3 \cdot b^{-3} + f_3 \cdot b^{-3} + \dots$$

Deduzimos então:

$$f \cdot b^3 - (c_1c_2c_3)_b = f_3 \in [0, 1[\qquad (c_1c_2c_3)_b = \lfloor f \cdot b^3 \rfloor$$

Os três algarismos c_1, c_2, c_3 podem ser recuperados como os algarismos associados a $\lfloor f \cdot b^3 \rfloor$ em base b, e a fração restante f_3 pode ser vista como a parte fracionária de $f \cdot b^3$.

Se em algum instante a fração obtida f_k for zero, então os algarismos associados a partir deste instante são todos 0 e diremos que o número fracionário f **tem uma representação exata através dum número finito de algarismos em base** b (na realidade continua a ter infinitos algarismos mas são todos eles 0 a partir dum ponto). Escrevemos então f com ajuda duma sequência finita dos correspondentes numerais $f = (0.c_1c_2 \dots c_k)_b$ (sendo $c_k \neq 0$ e entendemos que os algarismos posteriores são todos nulos)

A sequência de algarismos $(c_1, c_2, c_3, \dots)$ associados a um número fracionário f é por regra geral infinita, e é chamada **mantissa** do número f. Sendo $f > 0$ podemos ter a certeza de que $f > b^{-m}$ para algum m e portanto $f \cdot b^m > 0$, esta mantissa tem algum algarismo c_m não nulo. O primeiro valor $-m$ que satisfaz $f \geq b^{-m}$ é chamado **expoente do número** f, e o algarismo c_m associado o **algarismo mais significativo** do número fracionário. O expoente é portanto o número $-m$ que satisfaz $b^{-m} \leq f < b^{1-m}$

Devemos indicar que conhecer a mantissa não simplifica a escrita do número f. Nomeadamente dar a mantissa implica dar uma sequência infinita de algarismos. É por este motivo que números como π não podem ser representados (em forma exata) com a notação decimal clássica, por exemplo. Podemos no entanto escolher valores "arredondados":

Nota 1.6. Afirmar que em base b o número natural ou fracionário x tem expoente k, equivale a afirmar que $b^k \leq x < b^{k+1}$, ou que $k \leq \log_b x < k + 1k$ (onde usamos a função analítica de logaritmo em base b).

Definição 1.7. *Consideremos um número fracionário f com mantissa dada pela sequência infinita de algarismos $(c_1, c_2, c_3, \dots)$ Chamamos **arredondamento por corte ou por defeito** do número fracionário f com mantissa de p algarismos o número:*

$$\boxed{f^* = (0.c_1c_2 \dots c_p)_b = (c_1c_2 \dots c_p)_b \cdot b^{-p} = \sum_{i=1}^{p} c_i \cdot b^{-i}}$$

No arredondamento por corte sempre obtemos um valor $0 \leq f^* \leq f$, representável através de p algarismos em base b (este arredondamento por corte poderia até ser o valor 0)

*Chamamos **arredondamento por excesso** dum número fracionário f com mantissa de p algarismos o número*

$$\boxed{f^{**} = (1 + (c_1c_2 \dots c_p)_b) \cdot b^{-p} = b^{-p} + \sum_{i=1}^{p} c_i \cdot b^{-i}}$$

Este número satisfaz $f < f^{**} \leq 1$ (o arredondamento por excesso poderia até ser o valor 1)

Chamamos ***arredondamento ao mais próximo*** *dum número fracionário f o valor arredondado f^* ou f^{**} mais próximo de f. Se chamamos $f_p = b^p \cdot (f - f^*)$ a fração restante após o algarismo p-ésimo, o arredondamento ao mais próximo é f^* se $f_p < 1/2$ (o arredondamento seria feito por defeito), é f^{**} se $f_p > 1/2$ (o arredondamento seria feito por excesso), e no caso particular $f_p = 1/2$ (as duas opções iriam estar à mesma distância) por convenção podemos aceitar como arredondamento ao mais próximo o valor f^*.*

Exemplo

Determinemos a sequência (infinita) de algarismos em base $b = 8$ para o número $f = 1/17$. Para isto devemos iterar o produto com $b = 8$, determinando as partes inteiras e frações:

$(\times 8)$	(8/17)	(64/17)	(104/17)	(16/17)	(128/17)	(72/17)	
f=1/17 f_i	8/17	13/17	2/17	16/17	9/17	4/17	...
c_i	0	3	6	0	7	4	...

A sequência de algarismos associados a $1/17$, em base octal, seria portanto $(0, 3, 6, 0, 7, \ldots)$. O expoente associado a $f = 1/17$ em base octal é portanto -2.

Poderíamos representar na escrita, o valor arredondado por corte com 5 algarismos:

$$f^* = 8^{-5} \cdot (03607)_8 = (0.03607)_8$$

Ou se usamos trígrafos para passar a notação binária temos um valor arredondado com 15 algarismos binários:

$$f^* = (2^3)^{-5} \cdot (000\,011\,110\,000\,111)_2 = 2^{-15} \cdot (000011110000111)_2 = (0.000011110000111)_2$$

O primeiro algarismo binário não nulo aparece na posição quinta. O expoente binário do número $f = 1/17$ é -5. Mais concretamente, temos $2^{-5} = 1/32 \leq 1/17 < 2^{-4} = 1/16$

Como o resto octal associado ao algarismo quinto é $9/17 > 1/2$, o valor arredondado ao mais próximo com 5 algarismos octáis ou com 15 algarismos binários não é f^* senão que tem mais uma unidade:

$$f^{**} = 8^{-5} \cdot (1 + (03607)_8) = 8^{-5} \cdot (03610)_8$$

$$f^{**} = 2^{-15} \cdot (1 + (000011110000111)_2) = 2^{-15} \cdot (000011110001000)_2$$

O número $f = 1/17$ pode ser então representado de maneira arredondada como:

$$(0.03610)_8 = (0.000011110001000)_2$$

Finalmente, a partir da representação binária podemos dar também os primeiros algarismos da representação hexadecimal:

$$2^{-15} \cdot (000011110000111)_2 = 2^{-16} \cdot (000011110000111*)_2 = (2^4)^{-4} \cdot (0000\,1111\,0000\,111*)_{16}$$

onde o algarismo 16 binário não é conhecido (poderia ser 0 ou 1) e portanto o quarto algarismo hexadecimal não é conhecido (poderia ser E ou F). Os primeiros algarismos hexadecimais são no entanto conhecidos $0, F, 0$. O primeiro não nulo é o segundo algarismo. Portanto em base 16 a fração f tem expoente -2. O arredondamento por corte com 3 algarismos hexadecimais é $(0.0F0)_{16}$, e com 4 algarismos poderia ser $(0.0F0E)_{16}$ ou $(0.0F0F)_{16}$, dependendo de qual é realmente o algarismo binário na casa 16 da fração.

Nota 1.8. Se fixamos o número de algarismos das nossas representações numéricas, conseguimos ter um sistema que não permite representar em forma exata todos os valores numéricos, mas onde podemos representa-los com um conjunto limitado de símbolos, com um erro que podemos controlar. Do ponto de vista prático, aceitar este erro não é problemático, dado que, de facto, as medições realizadas em engenharia ou ciências não proporcionam informação exata e só resultam válidas para um determinado número de algarismos significativos.

Qualquer número natural pode ser representado nesta forma. No entanto, há números reais que não são representáveis em forma decimal finita (π ou $\sqrt{2}$ ou $1/3$, por exemplo). O facto de podermos representar todos os números naturais mas não todos os números reais não é um problema exclusivo do método posicional de representação dos números. O conjunto dos números reais não é numerável, portanto qualquer sistema de representação dos reais através duma sequência finita de símbolos extraídos dum alfabeto finito vai deixar, de forma inevitável, algum real sem representação possível. Não é possível escrever todos os números reais com um sistema de representação onde cada número se represente por uma coleção finita de numerais extraídas dum conjunto finito.

Nos computadores, a limitação antes indicada subsiste. Mais ainda, o nosso cérebro permite reservar (em teoria) uma quantidade indeterminada de espaço para tratar com números, o que permite pensar expressões com um número arbitrariamente grande de algarismos (não há um limite prefixado do comprimento do número) e portanto podemos trabalhar com qualquer número natural que possamos imaginar. Nos computadores isto supõe um gasto intolerável de recursos, nos sistemas de numeração mais utilizados em computadores reserva-se só uma quantidade predeterminada de memória para variáveis numéricas, e portanto geralmente estes só podem representar um número finito de números. Nem sequer os números naturais podem ser fielmente representados mais do que até ao grau de precisão que previamente exijamos.

Qualquer número real positivo é soma duma **parte inteira** e uma **parte fracionária**. Se combinamos a representação exata de números naturais com a representação aproximada de números fracionários, conseguimos representar um número real com notação posicional, no que chamamos **representação posicional em base b com vírgula fixa**:

Definição 1.9. *Se $x > 0$ é um número real com parte inteira n e fração f, chamamos **representação arredondada por corte do número x com p algarismos na fração** a sequência de numerais separados pela vírgula (ponto da fração):*

$$(a_k \dots a_2 a_1 a_0 \,.\, c_1 c_2 \dots c_p)_b$$

Onde $(a_k, \dots, a_1, a_0)$ são os algarismos associados à parte inteira, sendo $a_k \neq 0$, e onde $c_1, \dots, c_p$ são os primeiros p algarismos associados à fração.

Neste sistema de numeração a vírgula sempre se situa depois do algarismo menos significativo da parte inteira, separando este da parte fracionaria.

O zero e os números reais negativos saem do esquema até aqui apresentado. No caso do zero, rompemos o esquema segundo o qual a parte inteira devia começar com uma algarismo não nulo. Podemos representar o zero simplesmente através do símbolo numeral 0 associado.

No caso de números negativos x, podemos inserir um signo à frente (o traço "–" se o valor for negativo, e a cruz "+" se o valor for positivo) e representar a seguir o número positivo $|x|$ associado.

Definição 1.10. *Para um número real positivo $x > 0$, chamamos **expoente do número** em base b o único inteiro k para o qual $b^k \leq x < b^{k+1}$. Esta noção coincide com a noção de expoente já estudada se tratamos com um valor x natural (sem componente fracionária) ou se tratamos com um valor x fracionário (sem componente natural).*

Para um número real negativo $x < 0$, chamamos expoente do número em base b o expoente associado a $|x|$. O número 0 não tem expoente associado.

Quando um número x tem expoente e, a representação arredondada por corte do número x com p algarismos na fração é também chamada **representação arredondada por corte com $k + 1 + p$ algarismos significativos**.

Nota: Ainda que na na maior parte dos países europeus é habitual usar o símbolo de vírgula para separar parte inteira e fração, a sintaxe da maior parte das linguagens de programação reserva a vírgula a outro tipo de tarefas, resultando então conveniente usar o ponto para separar parte inteira e fração. Nesta disciplina, onde resulta necessário a utilização de ferramentas computacionais, evitaremos a confusão e iremos manter como critério permanente a utilização do ponto (e não a vírgula) na nossa notação numérica.

1.2 Notação científica em em ponto flutuante

Uma desvantagem do sistema de numeração com vírgula fixa é que há muitos números que não podem ser representados em forma exata, só arredondada, e que aqueles representáveis em forma exata podem exigir uma sequência larga de algarismos na sua representação.

Pensemos por exemplo no modelo conhecido das reações químicas, onde é sabido que um número determinado de átomos é trocado entre os reagentes para dar os compostos resultantes. Para poder transformar massas (peso dos reagentes) em números (ou moles) de átomos, é usado o número de Avogadro N_A, definido como o número de átomos em 12 gramas de Carbono-12 num estado ideal de repouso energético. Um valor proposto de Número de Avogadro conta os átomos num cristal cúbico de Carbono-12 com 84 446 888 átomos de lado, número que seria

$$N_A = 84446888^3 = 602\,214\,141\,070\,409\,084\,099\,072$$

Isto só poderia ser representado em notação binária se usamos 79 algarismos. Fixar assim o número de Avogadro é importante porque equivale então a fixar uma definição da unidade de massa (grama).

Na realidade os químicos desconhecem o valor exato do número de Avogadro, nem sequer concordariam em se a definição faz sentido com uma interpretação estrita, mas não é necessário concordar no valor exato de número de Avogadro, porque na prática é totalmente impossível contar uma quantidade tão grande de átomos, todos idênticos e numa situação ideal de repouso que nunca acontece. Nestes modelos ideais a experiência sempre mostra desvios que introduzem erros na contagem ou nas medições. Para efeitos de trabalho é suficiente com dizer que este número tem expoente decimal $e = 24$, e que os algarismos mais significativos são $(6, 0, 2, 2, 1, 4)$, por exemplo. Escrevemos então:

$$N_A = 6.02214E_{10}24$$

Ou seja $N_A = 6.02214 \cdot 10^{24}$. Este valor é um exemplo de utilização de notação científica com vírgula flutuante.

Um segundo exemplo: Antes do conhecimento dos relógios atómicos, a definição de segundo como uma fração do dia (período de rotação da terra) tinha o problema de nem todos os dias terem na realidade a mesma duração. Originalmente isto se resolveu ao usar o período de translação da terra (ano tropical), e indicando que entre 1750 e 1890 o período médio foi de 365 dias, 5 horas, 48 minutos e 46 segundos. O segundo fica assim definido como a fração 1/31556926 do ano tropical médio naqueles séculos. Esta fração em base 10 seria representada por: $0.000000031688764615412 79\ldots$

Resulta mais simples retirar os zeros e usar a fração $(3.168876461541279\ldots)E-8$ do ano tropical médio como definição de segundo. Até resulta interessante ficarmos só com um valor arredondado que utilize poucos algarismos significativos: o segundo seria a fração $3.169E-8$ do ano tropical médio entre 1750 e 1890.

Nota: A expressão E que aparece antes de indicar o expoente não tem relação nenhuma com a constante de Euler e. Assim $1.2E3$ é o número 1200, em nada relacionado com $1.2 \cdot \mathrm{e}^3$

A notação posicional de ponto fixo só é apropriada para números da ordem de $\pm b^m$ ou $\pm b^{-m}$, com m não demasiado grande. Isto é, para números $x \neq 0$ com $\log_b |x|$ pequeno. Quando este logaritmo resulta grande, será necessário introduzir muitos algarismos à esquerda ou muitos zeros à direita da vírgula para representar um valor aproximado deste. Para evitar o problema podemos estudar qual é a ordem de grandeza b^e do número, qual é o seu "expoente", e expressar o número como múltiplo de b^e.

Definição 1.11 (Sistema de numeração em notação científica normalizada). *Para qualquer número real $x \neq 0$ e qualquer base natural $b > 1$, diremos que x tem expoente $e \in \mathbb{Z}$ se $|x| \in [b^e, b^{e+1}[$. Neste caso $|x|/b^e \in [1, b[$ poderá ser escrito na forma $(a_0.a_1a_2a_e\ldots)_b$ com a_i uma sucessão de algarismos onde o primeiro deles é não nulo. Chamaremos representação do número x em notação científica em base b a seguinte:*

$$\boxed{x = \pm(a_0.a_1a_2\ldots)_b \cdot b^e}$$

onde ***o sinal*** $\pm$ *é positivo se* $x > 0$ *e negativo se* $x < 0$. ***O expoente*** e *é o único inteiro que verifica* $|x| \in [b^e, b^{e+1}[$, *e o algarismo à frente do ponto decimal (algarismo mais significativo) é* ***não nulo***.

Assim temos:

$$2.345E_{10}5 \;=\; 2.345 \cdot 10^5 \;=\; 234500 \qquad (4.3)_8E_82 \;=\; (4 + \frac{1}{8} \cdot 3) \cdot 8^2 = 256 + 24 = 280$$

Nota 1.12. O expoente pode ser calculado por ser $e \leq \log_b |x| = \ln|x| / \ln b < e + 1$. O número zero não tem logaritmo nem tem expoente associado em nenhuma base b.

Para representar um número $x \neq 0$ em notação científica normalizada em base b com p algarismos, arredondado por corte ou ao mais próximo, temos que:

- Identificar o seu sinal. Considerar $|x|$.
- Se x é um número fracionário, multiplicar com b até obter um número maior que 1. O número de vezes que multiplicamos determina o expoente em base b, e a parte inteira é o algarismo mais significativo. Se ficamos com a restante parte fracionaria e iteramos novamente o produto com b outras $p-1$ vezes determinamos os restantes $p-1$ algarismos. Para o arredondamento ao mais próximo, somamos uma unidade no último algarismo se a fração restante for superior a $1/2$.
- Se x tiver componente inteira, extrair os algarismos desta componente inteira através da divisão com resto entre b. O instante em que encontramos quociente zero determinará o expoente do número. Se o expoente for p ou superior (aparecem p ou mais algarismos já na parte inteira), ficar só com os p algarismos mais significativos. Para o arredondamento ao mais próximo, somar uma unidade no último algarismo se a fração restante for superior a $1/2$.
- Se nas operações anteriores encontramos que a parte inteira de x tem expoente inferior a p, tomar a parte fracionária de x e determinar os algarismos associados a esta parte, até completar o número p de algarismos desejados em total.Para o arredondamento ao mais próximo, somamos uma unidade no último algarismo se a fração restante for superior a $1/2$.

Exemplo

Representemos em notação científica octal, arredondado com 6 algarismos, o número $x = 28.31$.

Começamos pela parte inteira:

$(\div 8)$	$3 + \frac{4}{8}$	$0 + \frac{3}{8}$
$n = 28$ c_i	*3*	*0*
a_i	*4*	*3*

Assim $28 = (34)_8$. *Só temos 2 algarismos dos 6 que pretendíamos. Portanto passamos ao estudo da fração:*

$(\times 8)$	*2.48*	*3.84*	*6.72*	*5.76*	
f=0.31 f_i	*0.48*	*0.84*	*0.72*	*0.76*	...
c_i	*2*	*3*	*6*	*5*	...

Encontramos $0.31 = (0.2365\ldots)_8$. *Como a última fração é* $0.76 > 1/2$, *se queremos dar o arredondamento ao mais próximo devemos incrementar uma unidade neste último algarismo:*

$$28.31 \simeq (34.2366)_8 = 3.42366E_81$$

Aqui usamos E_8 *para indicar claramente que nos referimos ao expoente octal, e sendo automaticamente interpretados os algarismos com a base octal.*

Se quisermos em notação binária normalizada temos que $8^1 = (2^3)^1 = 2^{3\cdot 1} = 2^3$ *logo:*

$$28.31 \simeq (011\,.\,100\,010\,011\,110\,110)_2 \cdot 8^1 = (1.1100010011110110)_2 E_2 4$$

Vemos que o expoente octal multiplicado com 3 não vai dar exatamente o expoente binário. Isto porque o algarismo mais significativo octal (3) precisa de 2 algarismos binários para a sua representação e na notação científica normalizada só podemos admitir um algarismo não nulo antes da vírgula.

Finalmente se queremos passar a hexadecimal, devemos usar $2^4 = 16$ *e temos:*

$$28.31 = (1\,.\,1100\,0100\,1111\,0110)_2 \cdot 16^1 = (1.C8FA)_{16} E_{16} 1$$

Nota 1.13. Devemos advertir que se o número $x \neq 0$ está dado em notação científica numa base b, $x = \pm(a_0.a_1a_2\ldots)_b \cdot b^e$ e se pretendemos encontrar a sua representação em base B, a mantissa original $a_0.a_1a_2\ldots$ não pode ser usada diretamente. Não devemos representar o número $(a_0.a_1a_2\ldots)_b$ em base B. Temos que tomar como ponto de partida o número original $|x| = (a_0.a_1a_2\ldots)_b \cdot b^e$. Por exemplo, o número $(1.1)_2 \cdot 2^5$ será o valor $(1 + (1/2)) \cdot 32 = 48 = 4.8 \cdot 10^1$. Não serve de nada considerar que $(1.1)_2 = 1 + 1/2 = 1.5$. A mantissa de $(1.1)_2$ não tem nada a ver com a mantissa do número x dado.

Para dar o número x em notação científica em base B devemos primeiro encontrar o expoente, o valor $e \in \mathbb{Z}$ tal que $x \in [B^e, B^{e+1}[$. Depois, usaremos o valor $x/B^e \in [1, B[$ para dar a sua mantissa $\bar{a}_0.\bar{a}_1\bar{a}_2\ldots$, mantissa onde $\bar{a}_0$ pode ser obtida como a parte inteira, e a fração proporcionará os restantes algarismos: Com a fração multiplicada com B temos uma nova parte inteira $\bar{a}_1$ e uma nova fração. Multiplicamos esta novamente com B e obtemos na parte inteira o seguinte algarismo $\bar{a}_2$ e uma nova fração. Podemos continuar até obter o número de algarismos que quisermos.

Assim, por exemplo, ainda que o primeiro algarismo significativo em base 10 de $x = 7 \cdot 10^3$ é o algarismo 7, em octal este mesmo número não tem como primeiro algarismos octal o 7. Sendo um valor inteiro basta dividir entre 8 e determinar os restos:

$(\div 8)$	875	$109 + \frac{3}{8}$	$13 + \frac{5}{8}$	$1 + \frac{5}{8}$	$0 + \frac{1}{8}$
$n = 7000$ c_i	875	109	13	1	0
a_i	0	3	5	5	1

Portanto $7000 = 7E_{10}3$ em octal seria $(15530)_8$, e em notação científica octal normalizada $7000 = (1.553)_8 E_8 4$. O algarismo mais significativo octal é 1 e o expoente octal 4, sem relação nenhuma com o algarismo mais significativo 7 ou o expoente 3 decimal.

Definição 1.14. *Chamamos* ***sistema de números de ponto flutuante*** *com p algarismos em base b e expoente no intervalo $[e_{min}, e_{max}]$ o conjunto de todos os números representáveis na forma:*

$$\pm(a_0a_1\ldots a_{p-1}) \cdot b^{-p+1} \cdot b^e$$

com $e \in [e_{min}, e_{max}]$ valor inteiro, e onde $a_0, \ldots, a_p$ são algarismos da base b, sendo o primeiro deles não nulo.

$$\mathrm{FP}(b, p, e_{min}, e_{max}) = \left\{ \pm(a_0\,.\,a_1\ldots a_{p-1})_b \cdot b^e : \begin{array}{l} e_{min} \leq e \leq e_{max} \text{ inteiro} \\ a_0, a_1, \ldots, a_{p-1} \text{ algarismos da base } b \\ a_0 \neq 0 \end{array} \right\}$$

Estes números são todos da forma $\pm(a_0.a_1\ldots a_{p-1})_b \cdot b^e$. Portanto estamos a falar dos números da forma $\frac{k}{b^{p-1}} \cdot b^e$ onde $k \in [b^{p-1}, b^p[$ é um valor inteiro.

$$\mathrm{FP}(b, p, e_{min}, e_{max}) = \left\{ \pm k \cdot b^{e-p+1} : \begin{array}{l} e_{min} \leq e \leq e_{max} \text{ inteiro} \\ b^{p-1} \leq k < b^p \text{ inteiro} \end{array} \right\}$$

Temos por exemplo no caso $b = 2$, $p = 2$ dois valores possíveis para $2^1 \leq k < 2^2$, os números 2 ou 3 (em binário 10 e 11), e se $[e_{min}, e_{max}] = [-1, 2]$, temos 4 valores possíveis para $e - p + 1$, nomeadamente $-2, -1, 0, 1$:

$$\mathrm{FP}(2,2,-1,2) = \{\pm 2\cdot 2^{-2},\ \pm 3\cdot 2^{-2},\ \pm 2\cdot 2^{-1},\ \pm 3\cdot 2^{-1},\ \pm 2\cdot 2^{0},\ \pm 3\cdot 2^{0},\ \pm 2\cdot 2^{1},\ \pm 3\cdot 2^{1}\} =$$
$$= \{\pm 0.5,\ \pm 0.75,\ \pm 1,\ \pm 1.5,\ \pm 2,\ \pm 3,\ \pm 4,\ \pm 6\}$$

Vemos que são números com expoente binário entre -1 e 2, e que podem ser representados em forma exata com a notação científica binária com 2 algarismos.

Este sistema de numeração pode ser usado para representar em forma aproximada números reais não nulos x cujo expoente em base b esteja no intervalo $[e_{min}, e_{max}]$.

Se x é um número não nulo com expoente situado no intervalo $[e_{min}, e_{max}]$, chamamos **arredondamento ao zero** (ou por corte) do número x o valor $x^* \in \mathrm{FP}(e,p,e_{min},e_{max})$ cujo sinal e expoente coincidem com o de x, e cuja mantissa seja a sequência de algarismos associada ao valor $|x|/b^{e-p+1}$. Isto é:

$$x^* = s\cdot \lfloor |x|/b^{e-p+1} \rfloor \cdot b^{e-p+1}$$

onde s representa o sinal ± 1 de x, e e representa o expoente em base b de x.

Chamamos **arredondamento longe do zero** (ou por excesso) do número x o valor $x^{**} \in \mathrm{FP}(e,p,e_{min},e_{max})$ cujo sinal e expoente coincidem com o de x, e cuja mantissa seja a sequência de algarismos associada ao valor $|x|/b^{e-p+1}$ mais uma unidade. Isto é:

$$x^{**} = s\cdot \lfloor |x|/b^{e-p+1} + 1 \rfloor \cdot b^{e-p+1}$$

onde s representa o sinal ± 1 de x, e e representa o expoente em base b de x.

Há uma aproximação para $x = \pm\left((a_0\,.\,a_1\ldots a_{p-1})_b \cdot b^e + \epsilon\right)$ (onde $a_0 \neq 0$, $0 \leq \epsilon < b^{e-p+1}$) que produz erros menores: o **"arredondamento ao mais próximo com p algarismos"**, também chamado arredondamento simétrico, feito através da seguinte lei:

1. Se $x - x^* < (b/2)\cdot b^{e-p}$, diremos que $x^* = \pm(a_0\,.\,a_1\ldots a_{p-1})_b \cdot b^e$ é o valor obtido por arredondamento ao mais próximo (ou arredondamento simétrico) com p algarismos significativos (Isto e, faz-se simplesmente o arredondamento por corte com p algarismos).
2. Se $x^{**} - x > (b/2)\cdot b^{e-p}$, diremos que $x^{**} = \pm\left((a_0\,.\,a_1\ldots a_{p-1})_b + b^{-(p-1)}\right)\cdot b^e$ é o valor obtido por arredondamento ao mais próximo (ou arredondamento simétrico) com p algarismos significativos (Isto é, soma-se 1 no último algarismo e arredonda-se depois por corte com p algarismos).
3. No caso limite em que os dois valores indicados são exatamente $(b/2)\cdot b^{e-p}$, podemos escolher qualquer dos dois como arredondamento, mas para tentar repartir erros positivos e negativos por igual, é costume escolher o arredondamento x^* se a_{p-1} é par e x^{**} se a_{p-1} é ímpar (o valor arredondado teria o último algarismo par).

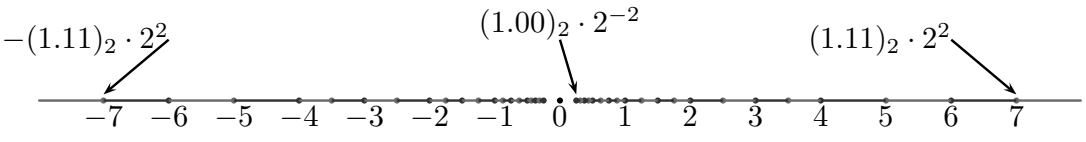

Figura 1.1. Arredondamento por corte em base $b = 2$ no terceiro algarismo. Para cada ponto, aparecem indicados os valores para os que é um arredondamento

Estes processos de arredondamento são úteis quando queremos encontrar um número em ponto flutuante x^* que sirva como aproximação dum número dado x. Se estamos a guardar a mantissa $a_0,\ldots,a_{p-1}$, qualquer destas técnicas produz arredondamentos x^* que têm p algarismos de precisão com respeito do valor real x.

Exemplo

Arredondamentos simétricos no sistema decimal até ao terceiro algarismo decimal:

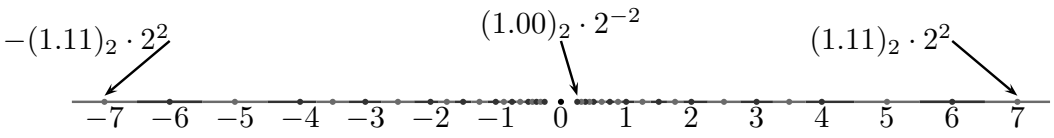

Figura 1.2. Arredondamento ao mais próximo em base $b = 2$ no terceiro algarismo. Para cada ponto, aparecem indicados os valores para os que é um arredondamento

número x	*Arredondamento* $\hat{x}$
0.2397	0.240
−0.2397	−0.240
0.23750	0.238
0.23650	0.236
0.23652	0.237
9.99991	10.0

Arredondamentos simétricos no sistema binário até ao sexto algarismo binário:

número x	*Arredondamento* $\hat{x}$
10.1110001100011	10.1110
0.0010010001100	0.00100100
1.1111111111110	10.0000
110.1010101010100	110.101
101.1011011000000	101.110
10.0100101000000	10.0101

Antes de terminar esta secção, deixemos um par de considerações acerca da representação de números em ponto flutuante:

1. É uma representação **finita**. Só um número finito de números podem ser representados em forma exata. A grande maioria dos números reais só podem representar-se em forma aproximada.
2. O conjunto de números representáveis em ponto flutuante é **mais denso na vizinhança do 0**.
3. **Existe um número positivo máximo** em ponto flutuante, sendo que os valores por cima deste valor não podem ser representados em forma arredondada.
4. **Existe um número positivo mínimo** em ponto flutuante, sendo que os valores situados entre 0 e este valor não podem ser representados em forma arredondada.
5. Há operações aritméticas que não têm um equivalente aceitável em ponto flutuante. Por exemplo, um número que possa ser representado em forma arredondada numa máquina binária pode ter um inverso que não se pode representar em forma arredondada.

A representação de números em ponto flutuante implica a necessidade de uma aritmética aproximada, uma aritmética própria para números em ponto flutuante.

Os números que satisfazem $|x| \geq b^{1+e_{max}}$ diremos que são um **overflow** para o sistema de numeração $\mathrm{FP}(b, p, e_{min}, e_{max})$

Os números que satisfazem $|x| < b^{e_{min}}$ diremos que são um **underflow** para o sistema de numeração $\mathrm{FP}(b, p, e_{min}, e_{max})$

Para poder executar operações aritméticas nestes números resulta conveniente estender com mais números.

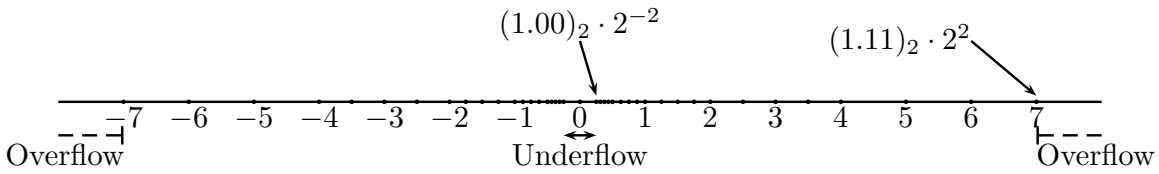

Figura 1.3. Valores representáveis em ponto flutuante em base $b = 2$, com $p = 3$ e expoentes máximo e mínimo ± 2

Definição 1.15. *Chamamos sistema numérico* ***estendido*** *normalizado* $\mathrm{FP}^*(b, p, e_{min}, e_{max})$, *com p algarismos em base b, com expoentes num intervalo* $[e_{min}, e_{max}]$ *o conjunto formado por* $\mathrm{FP}(b, p, e_{min}, e_{max})$, *e mais 5 elementos* $\{-\infty, +\infty, -0, +0, NaN\}$.

Os dois primeiros ("infinitos") para servir de arredondamento de valores negativos ou positivos que são overflow. Os dois seguintes ("zeros") para servir de arredondamento de valores negativos ou positivos que são underflow, ou para representar o zero. O último ("Not a Number") para ser usado como resultado de operações não válidas, por exemplo a divisão entre zero ou a soma de infinitos de sinal oposto.

Entenderemos que foi fixado algum sistema de arredondamento para números de expoente $e \in [e_{min}, e_{max}]$, e que este sistema determina uma função crescente, não contínua do conjunto dos números reais ao conjunto $\mathrm{FP}^*(b, p, e_{min}, e_{max})$. A função será chamada de **arredondamento em ponto flutuante** e denotada normalmente como $fl(x)$

Nota 1.16. A soma no sistema de números de ponto flutuante pode produzir números fora deste sistema. Assim por exemplo em base binária $b = 2$, com 2 algarismos $p = 2$, podemos representar $0.75 = (1.1)E - 1$ e podemos representar $1.5 = (1.1)E0$, mas a soma $2.25 = (10.01)_2 = (1.001)E1$ não seria representável.

A aritmética no conjunto $\mathrm{FP}^*(b, p, e_{min}, e_{max})$ faz-se de maneira arredondada. Para números deste sistema definimos uma **soma, produto, diferença e quociente arredondados**:

$$\boxed{x \oplus y = fl(x \cdot y) \quad x \ominus y = fl(x - y) \quad x \odot y = fl(x \cdot y) \quad x \oslash y = fl(x/y)}$$

No sistema numérico estendido a soma, produto com infinito, ou o quociente com infinito ou com zero seguem as regras conhecidas para o cálculo de limites. O resultado para operações que não fazem sentido desde este ponto de vista são arredondadas como NaN.

Devemos indicar que usar estas operações é complicado, as regras habituais deixam de ser válidas:

Podemos ter $x \oplus y = x$ sem ser $y = 0$, como no caso $x = 4$, $y = 0.75$ para $\mathrm{FP}(2, 2, -1, 2)$ com arredondamento por corte ao mais próximo.

Nota 1.17. Para conhecer de maneira precisa qual é o sistema numérico que estamos a usar, não é suficiente com conhecer o conjunto de números que existem no sistema (ou seja o conjunto $\mathrm{FP}(b, p, e_{min}, e_{max})$ ou o extendido), mas também resulta necessário conhecer como estamos a arredondar números que não são do sistema a números do sistema (a função fl, por exemplo o arredondamento por corte ou ao mais próximo), e como são executadas as operações de soma, diferença, produto ou quociente no conjunto (normalmente por arredondamento do resultado dessa operação nos números reais).

Chamaremos **sistema aritmético** $\mathrm{FP}(b, p, e_{min}, e_{max}, A)$ o sistema formado pelos números em ponto flutuante estendido $\mathrm{FP}^*(b, p, e_{min}, e_{max})$ e onde os arredondamentos e operações aritméticas são os definidos pelo arredondamento simétrico.

Chamaremos **sistema aritmético** $\mathrm{FP}(b, p, e_{min}, e_{max}, T)$ o sistema formado pelos números em ponto flutuante estendido $\mathrm{FP}^*(b, p, e_{min}, e_{max})$ e onde os arredondamentos e operações aritméticas são os definidos pelo arredondamento por corte ou truncatura.

Representação de números numa memória binária

Pensemos numa memória com **q espaços livres** onde cada espaço admite b estados diferentes.

Cada espaço de memória poderia codificar algarismos $0, 1, \ldots, b-1$. Podemos usar isto para guardar na memória números inteiros, positivos ou negativos, com ajuda da sua sequência de algarismos em base b

Exemplo

Se trabalhamos com inteiros sem sinal (positivos) a serem armazenados numa palavra de 2 bytes (1 byte=8 bits; 2 bytes permitem guardar $q = 16$ bits de informação binária), então o número $(16521)_{10}$ será guardado como **0100000010001001**, *já que 16521 é $(40211)_8$, e $(100000010001001)_2$.*

Se trabalhamos com inteiros com sinal em palavras de 2 bytes, o número -16521 é armazenado como:

* 1100000010001001 *se se utiliza o método de representação com sinal (primeiro bit para o sinal, e 15 bits para o número).*
* *Com notação de excesso em 2^{15}, 16521 é guardado como $2^{15}+16521$, isto é:* 1100000010001001, *e -16521 como $2^{15} - 16521$, isto é:* 0011111101110111
* *Com notação de excesso em $2^{15} - 1$, 16521 é guardado como* 1100000010001000 *e -16521 como* 0011111101110110. *É a mesma expressão que antes, a restar 1.*

A representação de números reais exigiria uma sucessão de infinitos algarismos. Para representar os números reais, escolhemos uma base b e consideramos a notação científica arredondada:

$$\pm a_0 \,.\, a_1 a_2 \ldots a_{p-1} \cdot b^e$$

onde $a_i \in \{0, 1, \ldots, b-1\}$, sendo $a_0 \neq 0$.

Pensémo-lo **normalizado**, isto é, com o expoente escolhido em forma que $a_0 \neq 0$. Isto sempre é possível quando o número real não é o 0. A informação a guardar é a seguinte:

- O sinal do número, se é positivo ou negativo
- A sequência de algarismos $a_0 a_1 \ldots a_{p-1}$, que pode interpretar-se como um inteiro positivo, e que se conhece como *mantissa*, de comprimento p (falamos em precisão p)
- O valor de e, número inteiro que pode ser positivo ou negativo, o *expoente*

Existem normas para a codificação desta informação de ponto flutuante nas memórias binárias dos computadores. A maior parte das normas seguem o standard determinado pela Institute of Electrical and Electronics Engineers (IEEE) nomeadamente com o seu IEEE Standard 754. Esta opção determina, para máquinas com memória binária (unidades de informação são bits com 2 estados que permitem portanto codificar os algarismos binários), como é codificado um número se temos a sua representação binária em ponto flutuante com p algarismos binários:

Na representação numérica binária com ponto flutuante, cada espaço de memória (bit) pode estar ativo ou inativo (1 ou 0), e utiliza-se:

1. 1 bit para o sinal
2. vários bits para o expoente
3. vários bits para a mantissa

A mantissa é sempre um número positivo, enquanto o expoente pode ser negativo. Podemos guardar a mantissa sem problemas através da sua representação binária. Mais ainda, como o primeiro algarismo é sempre 1, resulta desnecessário guardar este valor, e só precisamos $p - 1$ espaços de memória para guardar os algarismos $a_1 \ldots a_{p-1}$ que determinam a mantissa $a_0.a_1 \ldots a_{p-1}$.

Para codificar o expoente (inteiro positivo ou negativo, com sinal), temos que escolher qualquer dos métodos já vistos. No IEEE recomenda-se o método de excesso. se temos q bits diferentes para codificar o expoente, temos espaço para guardar 2^q diferentes valores.

- Reserva-se normalmente a sequência de bits de expoente $1\ldots1$ como código que indica que pretendemos representar $\pm\infty$ ou NaN (distinguimos um caso de outro com um código específico nos bits da mantissa).
- Reserva-se normalmente a sequência de bits de expoente $0\ldots0$ como código que indica que pretendemos representar ±0 ou determinados valores pequenos "de-normalizados" (distinguimos o caso com um código específico nos bits da mantissa)
- Qualquer outra sequência de bits é a sequência associada a um número inteiro (o expoente), guardado com excesso em N (ou seja, esta sequência de bits representam um valor $1 \leq k \leq 2^q - 2$ para indicar que temos um número de expoente $e = k - N$ (os valores $k = 0$ ou $k = 2^q - 1$ foram reservados acima).
  Normalmente o excesso no expoente é $N = 2^{q-1} - 1$, o qual permite representar expoentes desde $2 - 2^{q-1}$ até $2^{q-1} - 1$.

Em função de quantos bits pretendemos usar, as escolhas mais frequentes são as seguintes:

Bits totais	16	32	64	128
Bits sinal	1	1	1	1
Bits mantissa $(p-1)$	10	23	52	112
Bits expoente q	5	8	11	15
Excesso expoente (N)	15	127	1023	16383
$[e_{min}, e_{max}]$	[-14,15]	[-126,127]	[-1022,1023]	[-16382,16383]

Para transformar números decimais em números em standard IEEE754 há que seguir os seguintes passos:

1. Se o número é negativo
   a) Põe o bit do sinal em '1'.
   Se o número é positivo
   b) Põe o bit do sinal em '0'.
2. Converte o número a forma binária.
3. Escreve-o com notação científica em base 2, normalizado de forma que $a_0 = 1$ e com uma mantissa de p algarismos.
4. Toma o expoente e guarda-o nos q algarismos reservados ao expoente com o método do excesso de $2^{q-1}-1$. (Neste sistema o expoente 0 representa-se como $011\ldots1$). Se o expoente for superior a e_{max}, codifica o número como ∞. Se o expoente for inferior a e_{min}, codifica o número como 0
5. Toma os p algarismos da mantissa binária, ignora o 1 que aparece como mais significativo e insere os restantes no espaço reservado à mantissa.

Como amostra, pode-se comparar os seguintes resultados obtidos num Processador Pentium4, com sistema WindowsXP, através dum programa C++

```
0 00000000000 0000000000000000000000000000000000000000000000000000  x=0.
0 01111111111 0000000000000000000000000000000000000000000000000000  x=1.
0 10000000000 0000000000000000000000000000000000000000000000000000  x=2.
0 10000000000 1000000000000000000000000000000000000000000000000000  x=3.
0 10000000001 0000000000000000000000000000000000000000000000000000  x=4.
0 10000000001 0100000000000000000000000000000000000000000000000000  x=5.
0 10000000001 1000000000000000000000000000000000000000000000000000  x=6.
0 10000000001 1100000000000000000000000000000000000000000000000000  x=7.
0 10000000010 0000000000000000000000000000000000000000000000000000  x=8.
0 10000000010 0010000000000000000000000000000000000000000000000000  x=9.
0 01111111110 0000000000000000000000000000000000000000000000000000  x=0.5
0 01111111001 0100011110101110000101000111101011100001010001111011  x=0.02
0 01111111000 0100011110101110000101000111101011100001010001111011  x=0.01
0 01111110101 0000011000100100111011010010111100011010100111111100  x=0.001
0 01111110110 1001100110011001100110011001100110011001100110011010 x=0.003125
0 10000001001 0000000000000000000000000000000000000000000000000000  x=1024
0 10000001010 0000000000000000000000000000000000000000000000000000  x=2048
0 10000001011 0000000000000000000000000000000000000000000000000000  x=4096
0 10000001011 0000000000010000000000000000000000000000000000000000  x=4097
0 10000001011 0000000000001000000000000000000000000000000000000000  x=4096.5
0 10000001011 0000000000000100000000000000000000000000000000000000  x=4096.25
1 10000000010 0010000000000000000000000000000000000000000000000000  x=-9.
1 01111111110 0000000000000000000000000000000000000000000000000000  x=-0.5
1 01111111001 0100011110101110000101000111101011100001010001111011  x=-0.02
```

Pode-se observar como o sinal é guardado no bit mais significativo, os $q = 11$ bits seguintes são reservados ao exponente ($2^{11} = 2048$), guardado com excesso de 1023 (por isso $1 = 2^0$ tem $(01111111111)_2 = 1023$ no espaço do expoente), e os 52 bits restantes são reservados à mantissa ($p = 53$ é a precisão), onde o primeiro algarismo ($1.\cdots$) não é guardado, posto que sempre é 1 (salvo para o 0). Para o número 0., único com mantissa que não começa pelo 1, reserva-se a expressão cheia de 0's. Esta expressão deve interpretar-se como 0 e não como $1.0\ldots\cdot 2^{-1023}$.

Quando o expoente é $(00000000000)_2 = 0$, não deve tomar-se $0 - 1023 = -1023$ e interpretar-se como um número real de expoente 2^{-1023}. Além do número 0 antes indicado, este valor de expoentes está reservado no IEEE754 para indicar certos objetos de controlo, os "denormalized numbers". Por outro lado, quando o expoente é $(11111111111)_2 = 2047$, não deve tomar-se $2047 - 1023 = 1024$ e interpretar-se como um número real de expoente 2^{1024}. Este valor de expoente no IEEE754 está reservado para indicar certos objetos de controlo como $\pm\infty$ (quando a mantissa é $0\ldots0$), NaN ("not a number", quando a mantissa é não nula),

1.3 Aproximações e erros.

No trabalho com modelos matemáticos criados para representar situações reais as grandezas com que se trabalha não são precisas. A diferencia entre o valor numérico (o número) que usamos para representar aproximadamente a grandeza e o autêntico valor desta (valor que pode ser desconhecido) é o que se conhece como **o erro da aproximação**.

Os resultados nos nossos cálculos são afetados por vários tipos de erro. A interpretação correta do resultado exige então conhecer quais as possíveis origens e se há forma de evitar as causas destes erros.

Ao fazer um cálculo matemático (aplicar um procedimento algorítmico) a partir de dados, o resultado é suscetível de conter erros que se podem classificar em dois tipos:

1. O erro devido à falta de precisão dos dados de partida.
2. e o erro que o algoritmo produz quando faz o cálculo com dados exatos.

A falta de precisão nos dados de partida está provocada pela forma em que esses dados são obtidos e representados. Este erro tem uma **componente sistemática** e uma **componente aleatória**. A parte sistemática pode dever-se, por exemplo, a uma má calibração dos aparelhos de medição, ao arredondamento decimal que se faz nos dados, ou ao erro herdado duma série de operações feitas para gerar esses dados. A parte aleatória aparece quando os dados são obtidos através de um processo de tipo aleatório. Assim, uma mesma medição realizada em condições "idênticas" (até onde se possa controlar) é influenciada por fatores não controláveis, e produz resultados diferentes quando repetida, é uma experiência aleatória onde cada resultado individual está afetado por fatores aleatórios.

Os erros dos dados de partida podem também dever-se a que estes dados sejam o resultado da aplicação dum método numérico (que pode introduzir erros) desde valores conhecidos anteriormente (valores que podiam também conter erros)

Por outra parte, os algoritmos numéricos podem também produzir erros quando aplicados a dados exatos. Os resultados contêm então um erro que será propagado se se aplicam outros algoritmos. As fontes de erro mais comuns na aplicação de algoritmos são:

1. Não correspondência do modelo matemático com a realidade. As ciências oferecem-nos modelos que pretendem ser fiel reflexo da realidade. No entanto, isto não é assim, os modelos costumam ter um âmbito no qual se ajustam em forma aceitável à realidade e um âmbito no qual é preciso um modelo diferente para tratar o problema. Os algoritmos derivados destes modelos levam a resultados com maior ou menor erro em função de como se ajuste este modelo com a realidade.
2. Erros humanos e da máquina. O programador de algoritmos tem que ter um extremo cuidado na interpretação da teoria e na forma de representar esta no código que se programa ou no desenho de circuitos que se ocupam das operações internas. Pode acontecer, até com os melhores programadores e fabricantes, que estes nem sempre, frente a todos os dados, produzam os resultados esperados.
3. Erros de arredondamento. A causa deste erro é a impossibilidade de representar números reais numa memória finita. O erro de arredondamento aparece quando um número real é substituído por um valor expressado por um número fixo de algarismos (decimais ou binários). A limitação da aritmética de ponto flutuante leva a perda de informação que, dependendo do contexto pode ser mais ou menos importante. Como exemplo, a transformação de um número em ponto flutuante de base 10 a base 2 pode introduzir um erro de arredondamento: quase sempre que uma máquina começa tratar números que uma pessoa introduz, o primeiro que a máquina faz é adicionar um erro (o número é guardado como aquele valor em ponto flutuante em base 2 que melhor o representa).
4. Erros de truncatura. A causa deste erro é a impossibilidade de computar valores limite, que exigiriam tempo infinito para serem obtidos. Muitas ferramentas matemáticas usam a noção de limite duma sucessão. No entanto, a sucessão infinita não pode ser calculada explicitamente e o valor limite não pode ser obtido através da sucessão e da definição de limite. Programar um algoritmo que use um valor limite exige então truncar o cálculo dos termos da sucessão num determinado ponto em que espera-se ter obtido um valor o bastante próximo do valor limite, que não pode ser alcançado. Um exemplo grosseiro de truncatura é mudar uma determinada função $f(x)$ pelo polinómio de Taylor de segunda ordem associado $f(0) + f'(0) \cdot x + f''(0)/2 \cdot x^2$ (trunca-se a expressão infinita $f(x) = f(0) + f'(0) \cdot x + f''(0)/2 \cdot x^2 + f'''(0)/6 \cdot x^3 + \ldots$ de Taylor na segunda parcela). O erro de arredondamento na representação de ponto flutuante pode-se interpretar como um erro de truncatura ao substituir uma soma infinita $\sum_{k=1}^{\infty} a_{-k} b^{-k}$ pela soma truncada num passo dado.

Medidas de erro

Pensemos numa grandeza x que nós aproximamos pelo valor x^*. Na secção anterior demos vários motivos pelo qual o valor aproximado, no geral, pode não coincidir com o autêntico.

O valor x^* trata-se então duma aproximação. Esta questão apresenta-se quando trabalhamos com valores reais, mas também irá aparecer quando trabalharmos com pontos do plano $\mathbb{R}^2$, do espaço tridimensional $\mathbb{R}^3$ ou em geral com elementos de $\mathbb{R}^n$ (os elementos $x = (x_1, x_2, \ldots, x_n)$ sendo cada x_i um valor real)

Pretendemos distinguir quais são aproximações mais precisas e quais menos. Iremos associar a cada valor x e cada possível aproximação x^* um valor não negativo, uma **medida do erro** de x^* como aproximação de x.

Aparece uma dúvida natural, nomeadamente se consideramos $x^* = 2581$ como aproximação de $x = 2581 + \frac{2}{3}$, ou se consideramos $y^* = 1$ como aproximação de $y = 1 + \frac{1}{3}$, qual delas consideramos melhor aproximação? Por uma parte a distância entre x^* e x é $\frac{2}{3}$ (no primeiro caso), enquanto a distância entre y^* e y é $\frac{1}{3}$ (menor neste segundo caso). Por outra parte a quantidade $\frac{1}{3}$, em termos das unidades que estamos a usar, resulta maior do que $\frac{2}{3}$, em termos das unidades que estamos a usar.

Para medir o erro em $\mathbb{R}^n$ precisamos de uma noção de distância entre x^* e x (para o erro absoluto), e de uma forma de ponderar esta distância em termos da grandeza de x (para o erro relativo).Isto pode ser resolvido com ajuda da noção de **norma**:

Definição 1.18. *Uma **norma** $\|\cdot\|$ em $\mathbb{R}^n$ é uma função real $\mathbb{R}^n \to \mathbb{R}$ que satisfaz:*

1. *A norma é definido-positiva:* $\|x\| > 0, \quad \forall x \in \mathbb{R}^n, \quad x \neq (0, \ldots, 0)$
2. *A norma é homogénea:* $\|\lambda \cdot x\| = |\lambda| \cdot \|x\|, \quad \forall x \in \mathbb{R}^n, \forall \lambda \in \mathbb{R}$
3. *A norma é sub-aditiva:* $\|x + x'\| \leq \|x\| + \|x'\|, \quad \forall e, e' \in E$

Exemplos de normas são as seguintes:

Definição 1.19. *Seja $p \geq 1$ um número real. Chamamos **norma**-p dum ponto $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$ o valor:*

$$\boxed{\|x\|_p = \sqrt[p]{|x_1|^p + |x_2|^p + \ldots + |x_n|^p}}$$

*Observamos que para $p \to +\infty$, isto define uma **norma**-∞*

$$\boxed{\|x\|_\infty = \max(|x_1|, |x_2|, \ldots, |x_n|)}$$

No caso $n = 1$ (o conjunto dos números reais) estas normas são todas a mesma, o cálculo de $|x|$

No caso $n = 2$ (os pontos do plano), temos três normas importantes muito usadas:

$$\|(x_1, x_2)\|_1 = |x_1| + |x_2|, \quad \|(x_1, x_2)\|_2 = \sqrt{|x_1|^2 + |x_2|^2}, \quad \|(x_1, x_2)\|_\infty = \max(|x_1|, |x_2|)$$

Denominadas norma-1, norma-2 (ou Euclidiana) e norma-∞ no plano.

A norma Euclidiana é frequente, em particular por ter a sua origem na geometria Euclidiana, ainda que o seu cômputo supõe múltiplas operações. As normas $\|\cdot\|_\infty$ e $\|\cdot\|_1$ são frequentes pela sua simplicidade de cálculo e uso intuitivo.

A diferença entre dois pontos de $\mathbb{R}^n$ é um novo elemento $x - y \in \mathbb{R}^n$. A norma desde vetor é chamada a **distância entre os pontos**, com respeito da norma escolhida.

Definição 1.20. *Chamamos **produto escalar** $x \cdot y$ de dois elementos de $\mathbb{R}^n$ o seguinte valor real (escalar):*

$$(x_1, \ldots, x_n) \cdot (y_1, \ldots, y_n) = x_1 y_1 + x_2 y_2 + \ldots + x_n y_n$$

As seguintes propriedades são importantes para esta operação

Proposição 1.21. *O produto escalar em $\mathbb{R}^n$ satisfaz:*

- *É simétrico* $x \cdot y = y \cdot x$

- *É linear em cada componente, para escalares $\alpha \in \mathbb{R}$ e vetores $x, y, \bar{x}, \bar{y} \in \mathbb{R}^n$:*

$$(x + \bar{x}) \cdot y = x \cdot y + \bar{x} \cdot y, \quad x \cdot (y + \bar{y}) = x \cdot y + x \cdot \bar{y}$$

$$(\alpha x) \cdot y = \alpha(x \cdot y) = x \cdot (\alpha y)$$

- *Com respeito da norma euclidiana:*

$$|x \cdot y| \leq \|x\|_2 \cdot \|y\|_2, \qquad x \cdot x = \|x\|_2^2$$

- *Para vetores $u \in \mathbb{R}^n$ unitários em norma-p (aqueles com $\|u\|_p = 1$):*

$$\max_{\|u\|_p = 1} |x \cdot u| = \|x\|_q$$

sendo q o valor que satisfaz $\frac{1}{p} + \frac{1}{q} = 1$

Esta última propriedade permite afirmar que a norma-p tem uma **norma dual**, a norma-q, sendo que:

$$\boxed{|x \cdot y| \leq \|x\|_q \cdot \|y\|_p \qquad (\frac{1}{p} + \frac{1}{q} = 1)}$$

Esta desigualdade é precisa (em inglês "sharp"), no sentido de que para qualquer x é possível encontrar um y onde é satisfeita a igualdade, e reciprocamente.
(A norma dual da norma-2 é novamente a norma-2. A norma dual da norma-1 é a norma-∞. A norma dual da norma-∞ é a norma-1)

Mais uma desigualdade precisa é a seguinte:

$$\boxed{|x \cdot y| \leq |x_1| \cdot |y_1| + |x_2| \cdot |y_2| + \ldots + |x_n| \cdot |y_n|}$$

(a norma do produto escalar não é superior ao produto escalar das normas de cada componente)

Definição 1.22. *Se fixamos uma norma $\|\cdot\|$ em $\mathbb{R}^n$, chamamos **distância** associada a esta norma a função $\Delta(x, y) = \|x - y\|$. Esta função satisfaz as seguintes propriedades:*

- *É não negativa $\Delta(x, y) \geq 0$, com $\Delta(x, y) = 0 \Leftrightarrow x = y$*
- *É simétrica $\Delta(x, y) = \Delta(y, x)$*
- *Satisfaz a desigualdade triangular: $\Delta(x, y) \leq \Delta(x, z) + \Delta(z, y)$*

Distâncias muito usadas são a distância euclidiana Δ_2 (comprimento dum percurso que une dois pontos, se usamos qualquer direção retilínea), a distância Δ_1 chamada "taxicab" ou "Manhattan" (é o comprimento dum percurso entre os pontos, se só usamos direções paralelas aos eixos), e a distância Δ_∞, chamada também de Chebychev ou "do rei no xadrez".

Os conceitos de norma e de distância vão permitir a introdução duma **medida do erro**:

Definição 1.23. *Chamamos **medida do erro absoluto, com respeito duma norma** $\|\cdot\|$ em $\mathbb{R}^n$, dum ponto x^* como aproximação doutro ponto x o valor:*

$$\boxed{\Delta(x^*, x) = \|x^* - x\|}$$

*Se $x \neq (0, \ldots, 0)$, chamamos **medida de erro relativo** de x^* como aproximação de x, o valor:*

$$\boxed{\delta(x^*, x) = \frac{\Delta(x^*, x)}{\|x\|}}$$

*O produto do erro relativo com 100 é chamado **percentagem de erro** da aproximação.*

Como notação abreviada para indicar um erro absoluto ou um erro relativo, usamos as seguintes

$$x^* = x \pm \epsilon \Leftrightarrow \Delta(x^*, x) \leq \epsilon$$

$$x^* = x \cdot (1 \pm \alpha) \Leftrightarrow \delta(x^*, x) \leq \alpha$$

Uma utilidade da norma é que permite definir quando uma sucessão de pontos $x(1), x(2), x(3), \ldots$ é convergente a um determinado limite $a \in \mathbb{R}^n$.

Definição 1.24. *Diremos que uma sucessão de pontos $x(1), x(2), x(3), \ldots$ é **convergente** a um determinado ponto limite $a \in \mathbb{R}^n$ quando a sucessão de erros absolutos associada $\Delta(k) = \|x(k) - a\|$ tiver limite zero (equivalentemente, se a sucessão de erros relativos tiver limite zero). Escrevemos então*

$$\boxed{\lim_{k\to\infty} x(k) = a \qquad (\lim \Delta(k) = 0)}$$

*Diremos que há **convergência linear** se existe uma constante $c < 1$ para a qual*

$$\Delta(k+1) < c \cdot \Delta(k) \qquad (c < 1)$$

*Diremos que há **convergência quadrática** (mais rápida que a linear) se existe alguma constante c para a qual $\Delta(k+1) < c \cdot (\Delta(k))^2$, e em geral diremos que tem **convergência de ordem** q se existe uma constante c para a qual*

$$\boxed{\Delta(k+1) < c \cdot (\Delta(k))^q}$$

(A definição se trocamos Δ por δ resulta ser coincidente)

Por exemplo a sucessão $\frac{1}{2^k}$ tem convergência linear a $\overline{x} = 0$ Uma propriedade notável é que todas as normas em $\mathbb{R}^n$ são equivalentes, em qualquer questão relativa à convergência ou à ordem de convergência de sucessões. Se uma sucessão de pontos é convergente a um ponto a com ordem de convergência q quando usamos uma norma, a mesma propriedade continua a ser válida se usamos qualquer outra norma. Também resulta equivalente definir a convergência ou a ordem de convergência através do erro relativo.

Nota 1.25. Se pensamos em $\mathbb{R}$, onde a igualdade $\|x\| = a$ só tem como soluções $x = a$ ou $x = -a$, conhecer o erro relativo $\delta = \delta(x, x^*)$ permite escrever $x = x^* \cdot (1 \pm \delta)$. Podemos também escrever $x = x^* \pm \Delta$ (onde $\Delta = \Delta(x^*, x)$)

Os erros absolutos são claramente simétricos. Isto é, temos $\Delta(x^*, x) = \Delta(x, x^*)$. Os erros relativos no entanto, não sã simétricos. No caso da medição de erros relativos em $\mathbb{R}$ temos:

$$\delta(x, x^*) = \pm\frac{x^* - x}{x^*} = \pm\frac{x^* - x}{x}\left(1 + \frac{x^* - x}{x}\right)^{-1} = \frac{\pm\delta(x^*, x)}{1 \pm \delta(x^*, x)} = \pm\delta(x^*, x) - \delta(x^*, x)^2 + \ldots$$

(onde $\ldots$ representa o desenvolvimento de Taylor de $\frac{\delta}{1+\delta}$ para δ próximo de zero)

Portanto se $\delta(x^*, x)$ é pequeno o seu quadrado é muito menor e temos $\delta(x, x^*) \simeq \delta(x^*, x)$. As duas formas de definir o erro relativo são aproximadamente iguais, para valores do erro pequenos. Podemos assim escrever $x = x^* \cdot (1 \pm \delta)$ se conhecemos $\delta = \delta(x^*, x) \simeq \delta(x, x^*)$

Por outra parte, utilizaremos com frequência a notação $x^* \pm \epsilon$ como aproximação de x. Com esta expressão queremos dizer que existe um valor $e \in [-\epsilon, \epsilon]$ tal que $x^* = x + e$. Neste caso um valor aproximado para x é x^* e o valor absoluto do erro cometido na aproximação é $|x^* - x| \leq \epsilon$.

Quando trabalhamos em $\mathbb{R}$ há uma forma de indicar se o erro relativo é grande. Esta é a noção de **número de algarismos significativos da aproximação**. Suponhamos que temos um valor x desconhecido cuja representação em notação científica em base b é:

$$x = \pm(a_0.a_1a_2 \ldots a_{p-1} \ldots)_b \cdot b^e$$

com $a_0 \neq 0$. Isto é, temos $b^e \leq x < b^{e+1}$. Suponhamos que temos um valor x^* como aproximação de x.

Definição 1.26. *Dado $b^e \leq x < b^{e+1}$, diremos que x^* aproxima x* ***com*** *p* ***algarismos significativos*** *ou* ***com*** *p* ***algarismos de precisão*** *em base b se:*

$$\boxed{\left|\frac{x^* - x}{b^e}\right| \leq \frac{b}{2} \cdot b^{-p}}$$

quando b é par, isto quer dizer que:

$$|x^* - x| \leq (0.00\ldots\overbrace{0}^{p-1}\overbrace{(b/2)}^{p})_b \cdot b^e$$

Por exemplo a aproximação 0.008234 ± 0.000004 dum número x desconhecido, podemos afirmar que é uma aproximação com 3 algarismos significativos: $x \in [0.00823, 0.008238] \subset [10^{-3}, 10^{-2}]$ (portanto $e = -3$) e $|x^* - x|/10^{-3} \leq 0.004 \leq 5 \cdot 10^{-3}$. No entanto, 0.008234 ± 0.000006 não podemos saber se é uma boa aproximação com 3 algarismos significativos, o máximo que podemos afirmar é que tem 2 algarismos significativos:

$$\left|\frac{x^* - x}{10^{-3}}\right| \leq 0.006 \begin{cases} \not\leq 5 \cdot 10^{-3} \\ \leq 5 \cdot 10^{-2} \end{cases}$$

dependendo de qual é o valor real desconhecido, pode ser que a diferença alcance até 0.006, que não é menor que $5 \cdot 10^{-3}$ mas sim é menor do que $5 \cdot 10^{-2}$. é uma aproximação com 2 algarismos significativos.

O número de algarismos significativos numa aproximação é uma forma de expressar o erro relativo cometido na aproximação.

Porquê tomamos $b/2 \cdot b^{-p}$ para falar de algarismos significativos duma aproximação?. Isto é devido ao método de arredondamento:

Nota 1.27. Seja x um valor real não nulo. Ao determinar os seu primeiros p algarismos significativos em base b e a fração restante temos

$$x = \pm\left((a_0\,.\,a_1\ldots a_{p-1})_b \cdot b^e + \epsilon\right)$$

com $a_0 \neq 0$, $0 \leq \epsilon < (0.0\ldots\overbrace{1}^{p-1})_b \cdot b^e = b^{e-p+1}$.

O valor $x^* = \pm(a_0\,.\,a_1\ldots a_{p-1})_b \cdot b^e$ era a aproximação de x obtida por **arredondamento por corte com p algarismos** ("aproximação ao zero", "por corte" ou "por defeito"). O valor $x^{**} = x^* + (0.0\ldots\overbrace{1}^{p-1})_b \cdot b^e = x^* + b^{e-p+1}$ é o **arredondamento "por excesso" com p algarismos**.

Podemos observar então:

$$\frac{|x^{**} - x^*|}{b^e} = b^{-p+1}$$

Como x está situado entre o arredondamento por corte x^* e o arredondamento por excesso x^{**}, podemos afirmar que nestes dois métodos de arredondamento temos:

$$\frac{|fl(x) - x|}{b^e} \leq b^{-p+1} \quad \text{por corte ou por excesso}$$

Por outra parte seja qual for x, ao considerar o mais próximo destes dois arredondamentos iremos ter o arredondamento ao mais próximo. Para este arredondamento portanto:

$$\frac{|fl(x) - x|}{b^e} \leq \frac{1}{2} b^{-p+1} \text{arredondamento simétrico}$$

No **arredondamento simétrico com p algarismos** em base b, podemos ter a certeza que o valor arredondado $\hat{x}$ representa uma **aproximação de x com p algarismos significativos**.

O erro relativo numa aproximação feita por arredondamento fica então limitado pelo número de algarismos usados no arredondamento. Assim o espaço de memória que utilizarmos para guardar mantissa dum número influencia o erro relativo que admitimos cometer nos arredondamentos.

O espaço de memória que usamos para guardar o expoente influencia a diversidade de expoentes que conseguimos guardar.

Reservar muito espaço para os expoente irá permitir guardar de maneira arredondada um intervalo maior de números, e reservar muito espaço para a mantissa irá permitir que estes números sejam guardados com maior precisão (menor erro relativo). A metodologia do IEEE fixa um compromisso entre precisão e amplitude do sistema números usados.

Nota 1.28. Medir a velocidade de convergência duma sucessão através da "ordem de convergência" pode interpretar-se como uma maneira de medir quantos algarismos significativos ganhamos em cada novo termo da sucessão.

Podemos visualizar graficamente se uma sucessão é convergente com ordem de convergência q, e interpretar o significado em termos dos algarismos significativos de precisão de cada um dos termos como aproximação do valor limite.

Consideremos uma sucessão $(x(k))$ convergente a um valor $\overline{x} > 0$, e que o expoente binário de k é o valor $e \in \mathbb{Z}$, isto é, $\overline{x} \in [2^e, 2^{e+1}[$. O número de algarismos binários de precisão de $x(k)$ como aproximação de $\overline{x}$ está dada por $prec(k) = -\log_2 \frac{|x(k)-\overline{x}|}{2^e} = e - \log_2 \Delta(k)$.

A condição $\Delta(k+1) < \text{cte} \cdot (\Delta(k))^q$ equivale a $\log_2 \Delta(k+1) < \log_2 \text{cte} + q \log_2 \Delta(k)$, portanto a $e - \log_2 \Delta(k+1) > -\log_2 \text{cte} + (1-q)e + q(e - \log_2 \Delta(k))$, isto é, o número de algarismos binários de precisão $prec(k)$ devem verificar $prec(k+1) > \alpha + q \cdot prec(k)$. A cada novo termo da sucessão, o número de algarismos de precisão fica multiplicado com q (salvo um termo α somado, que será pouco relevante se o número de algarismos significativos for já elevado)

Portanto um método para calcular a ordem de convergência duma sucessão seria calcular os pontos $(prec(k), prec(k+1))$. Se estes pontos podem ser situados num semiplano $y > \alpha + qx$ limitado inferiormente por uma reta de declive q, a sucessão tem ordem de convergência q.

Da mesma maneira, com ajuda dos erros absolutos: se os pontos $(\log \Delta(k), \log \Delta(k+1))$ ficam situados num semiplano $y < q \cdot x$ limitado superiormente por uma reta de declive q, a sucessão tem ordem de convergência q. Podemos estudar assim a ordem de convergência se representamos graficamente os pontos $(\log_2 \Delta(k+1), \log_2 \Delta(k))$.

Nota 1.29. Para o arredondamento por corte $a^* = fl(a)$ dum número $a \neq 0$ numa máquina binária com precisão p temos $\delta(a^*, a) \leq 2^{-p}$. Portanto no caso dum elemento $a \in \mathbb{R}^n$ as componentes (valores reais) são aproximadas com $|a_i^* - a_i| \leq 2^{-p} \cdot a_i$.

Em muitas situações queremos representar no computador não um número, senão um ponto, uma sequência $a = (a_1, \ldots, a_n)$ com n componentes reais. O procedimento mais normal é usar um conjunto de n posições de memória e introduzir em cada uma delas o valor arredondado a_i^*. Com a medição de erros através da norma infinito temos:

$$\delta_\infty(a^*, a) = \frac{\|a^* - a\|_\infty}{\|a\|_\infty} = \frac{\max|a_i^* - a_i|}{\max|a_i|} \leq \frac{\max|2^{-p} \cdot a_i|}{\max|a_i|} = 2^{-p} \cdot \frac{\max|a_i|}{\max|a_i|} = 2^{-p}$$

Portanto do ponto de vista da norma-∞, numa máquina com representação de números em ponto flutuante com p algarismos de mantissa, os vetores podem ser representados com erros relativos não superiores a 2^{-p}

Definição 1.30. *Chama-se* ***unidade de arredondamento*** *duma máquina o supremo dos erros relativos introduzidos pelos arredondamentos existentes na máquina. Isto é, a unidade de arredondamento é o menor valor u que verifica:*

$$\boxed{\frac{|fl(x) - x|}{|x|} < u}$$

nos diferentes valores x que não sejam overflow/underflow.

Numa máquina binária com precisão p e que faz arredondamentos ao mais próximo, a unidade de arredondamento é então $u = 2^{-p}$.

Outro valor que produz informação análoga à precisão da máquina é o conhecido como épsilon da máquina.

Definição 1.31. *Chama-se **épsilon da máquina** num sistema de cômputo em ponto flutuante o menor número positivo ϵ representável na máquina e tal que:*

$$\boxed{1 \oplus \epsilon > 1}$$

onde $1 \oplus \epsilon$ quer dizer a soma de 1 e ϵ tal como feita pela máquina para números em ponto flutuante, normalmente $fl(1+\epsilon)$.

1.4 Propagação do erro

A noção de valor aproximado pode entender-se também para funções aproximadas ou algoritmos.

Pensemos num problema matemático com solução perfeitamente determinada mas dependente de parâmetros de entrada. Por exemplo, pensemos no problema de encontrar a menor das raízes reais do polinómio $x^2 + b \cdot x + c$. Sabemos que existe uma solução deste problema, e que está determinada pela aplicação da fórmula resolvente:

$$\text{Menor raiz de } x^2 + b \cdot x + c \text{ é } z = \frac{-b - \sqrt{b^2 - 4c}}{2}$$

A solução do problema matemático está assim descrita por $z = f(b, c)$ onde f é uma função nos parâmetros b, c que determinam o problema.

Se temos um sistema numérico com finitos números e onde existe um procedimento para somar, multiplicar, dividir, e extrair raízes destes números em forma aproximada, poderíamos criar um algoritmo f^* como aproximação da fórmula resolvente, ou seja uma série de instruções que:

- Precisa da introdução de dois valores b^*, c^* existentes no sistema numérico.
- Através das instruções programadas, produz como resultado um valor f^*
- O valor f^* devolvido irá ser usado como aproximação de $f(b, c)$, sempre que (b^*, c^*) sejam considerados aproximação de (b, c)

Assim o problema matemático tem parâmetros de entrada x (no caso anterior $x = (b, c)$), e conhecido o parâmetro existe uma solução determinada (aqui $f(x) = \frac{-b-\sqrt{b^2-4c}}{2}$).

No computador temos um algoritmo $f^* \colon x^* \mapsto y^*$ que admite como parâmetros de entrada valores x^* representáveis no computador e que consegue devolver um valor $y^* = f^*(x^*)$ representável no computador.

Muitas questões na matemática exigem a utilização de determinados valores (parâmetros) Suponhamos que temos um problema a resolver, que este problema depende de parâmetros numéricos e que este problema é resolvido através da função $f \colon x \to f(x)$. Suponhamos que conhecemos um algoritmo numérico $f^* \colon x^* \mapsto f^*(x^*)$.

Existem duas questões pertinentes para ver se f^* é bom para representar f:

- Se tivéssemos parâmetros representáveis em forma exata no computador $x = x^*$, será que o valor devolvido pelo algoritmo $f^*(x^*)$ é a melhor aproximação existente de $f(x^*)$? Será que o computador não consegue distinguir $f(x^*)$ de $f^*(x^*)$? Ou seja: $f^*(x^*) = fl(f(x^*))$ para qualquer x^* do sistema numérico usado?

- Se consideramos a solução $f^*(x^*)$, será que esta solução resulta ser a solução exata $f(x)$ para algum valor concreto x que o computador não distingue de x^*? Ou seja: $x^* = fl(x)$ para algum x que satisfaz $f(x) = f^*(x^*)$?

A primeira questão está referida a uma análise do erro direta para o algoritmo f^*. A segunda questão está referida a uma análise do erro inversa para o algoritmo f^*.

Em qualquer dos casos indicados, devemos ficar satisfeitos. O sistema numérico do computador não vai permitir, por muito bom que for o algoritmo, encontrar melhor representação de f do que a dada através de f^*

Se partimos dum valor numérico x^* e dum algoritmo f^*, tomar $f^*(x^*)$ como aproximação de $f(x)$ contem um erro $\Delta(f^*(x^*), f(x))$. Podemos decompor este em duas partes, uma devida à aproximação de a e outra à aproximação de f:

$$\boxed{(f^*(a^*) - f(a)) = \underbrace{(f^*(a^*) - f(a^*))}_{(1)} + \underbrace{(f(a^*) - f(a))}_{(2)}}$$

Nesta soma a primeira parte conhece-se como **erro de computação associado ao algoritmo** (erro devido a que o algoritmo não produz $f(a^*)$ nem quando $a^* = a$), e a segunda parte como **erro propagado por** f (erro não relacionado com o algoritmo f^* senão com o comportamento de f ao usar um valor aproximado de partida).

Definição 1.32. *Diremos que uma função f de variável real x é uma **função bem condicionada** no cálculo de f no valor a se*

$$\Delta(a^*, a) \text{ pequeno} \Rightarrow \Delta(f(a^*), f(a)) \text{ pequeno}$$

Do ponto de vista do programador, este pode tratar de criar um algoritmo estável para o cálculo de f, manter sob controlo o erro de computação através duma boa escolha do algoritmo. Por outra parte o erro propagado não depende dele, senão da função matemática $f(x)$ dada no modelo matemático, e do valor aproximado a^* que pode ter erros com respeito do valor exato a.

Um elemento destacado no estudo do erro propagado é a noção de **aproximação linear** duma função $f(x)$ num ponto $a \in \mathbb{R}^n$.

Definição 1.33. *Consideremos um ponto $a \in \mathbb{R}^n$ e uma função $f(x)$ definida em todos os pontos de $\mathbb{R}^n$ que satisfazem $\|x - a\| \leq \epsilon$ (bola de raio ϵ centrada em a). Consideremos as possíveis **funções afins** em $\mathbb{R}^n$ (funções do tipo $p(x) = p(x_1, \ldots, x_n) = c_0 + c_1x_1 + c_2x_2 + \ldots + c_nx_n$, polinomiais de grau 1). Diremos que $p(x)$ é a **linearização de** $f(x)$ **no ponto** a se $f(a) = p(a)$ e todas as sucessões $x(k) \neq a$ com limite a satisfazem:*

$$\lim_{k\to\infty} \frac{p(x(k)) - f(x(k))}{\|x(k) - a\|} = 0$$

*Neste caso diremos que $f(x)$ é **diferenciável no ponto** a, sendo $p(x)$ a sua aproximação linear.*

A condição do limite acima é uma forma de dizer que $p(x)$ é similar a $f(x)$. Indicaremos neste caso $f(x) \simeq p(x)$, ou com maior precisão:

$$f(x) = p(x) + o(x - a)$$

Definição 1.34. *Escrever $f(x) = g(x) + o((x-a)^q)$ irá significar, no sucessivo, que $f(a) = g(a)$ verificando ainda as sucessões $x(k) \neq a$ convergentes em a a seguinte propriedade:*

$$\lim_{k\to\infty} \frac{p(x(k)) - f(x(k))}{\|x(k) - a\|^q} = 0$$

A condição imposta exige em particular $p(a) = f(a)$. A linearização $p(x)$ tem assim a propriedade de coincidir com $f(x)$ no ponto a, e ainda se nos aproximarmos de a (através duma sucessão $x(k)$ de limite a), o erro de $p(x)$ como aproximação de $f(x)$ é pequeno (inferior a qualquer constante escolhida, multiplicada com $\|x(k) - a\|$).

Proposição 1.35. *Se $p(x)$ é a linearização de $f(x)$ no ponto a, então:*

$$\boxed{p(x) = f(a) + \partial_1 f(a) \cdot (x_1 - a_1) + \partial_2 f(a) \cdot (x_2 - a_2) + \ldots + +\partial_n f(a) \cdot (x_n - a_n)}$$

onde

$$\partial_i f(a) = \lim_{h \to 0} \frac{f(a_1, \ldots, a_i + h, \ldots, a_n) - f(a)}{h}$$

é chamada a ***derivada parcial da função f, na componente i, no ponto a.***

Prova: Consideremos o ponto $a = (a_1, \ldots, a_n)$. Resulta simples ver que qualquer função afim pode ser escrita na forma: $p(x) = c_0 + c_1 \cdot (x_1 - a_1) + \ldots + c_n \cdot (x_n - a_n)$, para alguma escolha de constantes c_i.

Tratemos de determinar as componentes $c_0, c_1, \ldots, c_n$ no caso em que $p(x)$ seja linearização de $f(x)$ no ponto a. Como exigimos $f(a) = p(a)$ temos necessariamente $c_0 = f(a)$

Queremos ainda provar que que $c_i = h_i'(a_i)$, onde usamos a função real de variável real $h_i(t) = f(a_1, \ldots, t, \ldots a_n)$, cuja derivada em a_i é precisamente a definição de $\partial_i f(a)$

Fixamos nossa atenção num i concreto. Provar que $h_i'(a_i) = c_i$ é o mesmo que provar

$$\lim_{t \to a_i} \frac{h_i(t) - h_i(a_i)}{t - a_i} = c_i$$

ou seja provar que para cada sucessão $t(k) \neq a_i$ convergente em a_i temos:

$$\lim_{k \to \infty} \frac{h_i(t(k)) - h_i(a_i)}{t(k) - a_i} = c_i$$

Provar esta igualdade não é difícil. Tomamos a sucessão de pontos $x(k) = (a_1, \ldots, t(k), \ldots, a_n)$, todos com a mesma componente em cada posição, salvo no lugar i.Temos neste caso $\|(x(k) - a)\| = \|(0, \ldots, t(k) - a_i, 0, \ldots, 0)\| = |t(k) - a_i| \cdot \|(0, \ldots, 1, 0, \ldots, 0)\| = |t(k) - a_i| \cdot s_i$ onde $s_i \neq 0$ é uma constante, a norma do vetor $(0, \ldots, 1, 0, \ldots, 0)$.

Temos em particular que $x(k)$ vai ser uma sucessão que nunca toma valor a (porque $t(k) \neq a_i$), mas tem limite a (porque $|t(k) - a_i|$ tem limite 0)

Mais ainda, devido à definição de h_i temos $f(x(k)) = h_i(t(k))$, $p(x(k)) = f(a) + c_i \cdot (t(k) - a_i) = h_i(a_i) + c_i \cdot (t(k) - a_i)$ e portanto como p é a linearização de f no ponto a temos:

$$0 = \lim_{k \to \infty} \frac{p(x(k)) - f(x(k))}{\|x(k) - a\|} = \lim_{k \to \infty} \frac{h_i(a_i) + c_i \cdot (t(k) - a_i) - h_i(t(k))}{|t(k) - a_i| \cdot s_i} =$$
$$= \lim_{k \to \infty} \left(c_i - \frac{h_i(t(k)) - h_i(a_i)}{t(k) - a_i} \right) \cdot \frac{t(k) - a_i}{|t_k - a_i| \cdot s_i}$$

Tendo em conta que o termo a multiplicar à direita é sempe $\frac{\pm 1}{s_i}$, o limite indicado só pode ser zero se as sucessões $t(k)$ indicadas satisfazem:

$$\lim_{k \to \infty} \frac{h_i(t(k)) - h_i(a_i)}{t(k) - a_i} = c_i$$

o qual é precisamente a definição para $\lim_{t \to a_i} \frac{h_i(t) - h_i(a_i)}{t - a_i} = c_i$, cada valor c_i é assim a derivada no ponto $t = a_i$ da função $h_i(t)$, como indicava a proposição. □

Nota 1.36. As derivadas parciais podem existir e no entanto o polinómio $p(x)$ indicado não satisfazer a condição para ser uma linearização de $f(x)$, ou seja, pode que a função não seja diferenciável. No entanto é sabido que quando as derivadas parciais existem em todos os pontos (não só em a), e são contínuas, o polinómio $p(x)$ indicado satisfaz sim a condição para ser uma linearização de $f(x)$ no ponto a, sendo portanto f diferenciável no ponto a.

Nota 1.37. A derivada parcial $\partial_i f(a)$ também é representada por $\frac{\partial f}{\partial x_i}(a)$, e coincide com o valor da derivada em $a_i \in \mathbb{R}$ da função real $f(a_1, \ldots, a_{i-1}, t, a_{i+1}, \ldots, a_n)$ na variável real t.

O cálculo da derivada parcial na componente i num ponto $(a_1, \ldots, a_n)$ exige assim considerar **quase todos** os valores $a_1, \ldots, a_n$ como **parâmetros fixos**, e o valor a_i como único variável, determinando então a derivada nesta variável. Podem assim aplicar-se todas as técnicas de derivação de funções reais de variável real.

Por exemplo, para $f(b,c) = \frac{-b-(b^2-4c)^{1/2}}{2}$, a derivada parcial na componente b seria:

$$\frac{\partial f}{\partial b}(b,c) = \frac{-1 - b \cdot (b^2-4c)^{-1/2}}{2} = \frac{-b - \sqrt{b^2-4c}}{2\sqrt{b^2-4c}}$$

e na componente c seria:

$$\frac{\partial f}{\partial c}(b,c) = \frac{2 \cdot (b^2-4c)^{-1/2}}{2} = \frac{1}{\sqrt{b^2-4c}}$$

O conjunto das derivadas parciais pode ser recolhido numa sequência

$$\boxed{\nabla_a f = (\partial_1 f(a), \partial_2 f(a), \ldots, \partial_n f(a)) \in \mathbb{R}^n}$$

o chamado **vetor gradiente de f no ponto a.**

Se conhecemos $f(a)$ e $\nabla_a f$, a linearização de f no ponto a é a seguinte:

$$p(x) = f(a) + (\nabla_a f) \cdot (x-a), \qquad f(x) = f(a) + (\nabla_a f) \cdot (x-a) + o(x-a)$$

Definição 1.38. *Diz-se que **um problema que depende de n parâmetros** e que é resolvido por uma função $f\colon \mathbb{R}^n \to \mathbb{R}$ está **bem condicionado** se a função f estiver bem condicionada, se variações pequenas nos dados introduzidos levam a variações pequenas nos resultados obtidos:*

Desde o ponto de vista do erro absoluto:

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

O coeficiente com que esta noção de "pequeno" é medida é o valor K que satisfaz:

$$|f(a^*) - f(a)| < K \cdot \|a^* - a\|$$

para valores de $\|a^* - a\|$ pequenos.

Desde o ponto de vista do erro relativo:

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

que também pode ser medido com um valor κ que satisfaz:

$$\frac{|f(a^*) - f(a)|}{|f(a)|} < \kappa \cdot \frac{\|a^* - a\|}{\|a\|}$$

para valores de $\delta = \frac{\|a^* - a\|}{\|a\|}$ pequenos.

Pensemos num ponto a e em todas as aproximações a^* onde o erro absoluto seja $\epsilon = \Delta(a^*, a) > 0$. Estas aproximações são todos os pontos da forma $a^* = a + \epsilon \cdot u$ com u unitário. Então se substituimos f pela sua linearização no ponto a temos:

$$\Delta(f(a^*), f(a)) \simeq |f(a) + (\nabla_a f) \cdot (a^* - a) - f(a)| = \epsilon \cdot |(\nabla_a f) \cdot u|$$

Se agora aplicamos a propriedade $|x \cdot y| \leq \|x\|_p \cdot \|y\|_q$ e tirando os termos quadráticos em ϵ, o máximo erro que podemos cometer é:

$$\begin{array}{c} \Delta_2(a^*, a) = \epsilon \Rightarrow |f(a^*) - f(a)| \leq \epsilon \cdot \|\nabla_a f\|_2 \\ \Delta_1(a^*, a) = \epsilon \Rightarrow |f(a^*) - f(a)| \leq \epsilon \cdot \|\nabla_a f\|_\infty \\ \Delta_\infty(a^*, a) = \epsilon \Rightarrow |f(a^*) - f(a)| \leq \epsilon \cdot \|\nabla_a f\|_1 \end{array}$$

(estas desigualdades são salvo termos em ϵ^2)

Pensemos agora em $a \neq 0$ com $f(a) \neq 0$ e nas aproximações a^* onde o erro relativo seja $\delta = \delta(a^*, a) > 0$. Nestas aproximações o erro absoluto é $\delta \cdot \|a\|$ e portanto estamos a falar de pontos da forma $a^* = a + \delta \cdot \|a\| \cdot u$ com u unitário. Novamente se substituímos f pela sua linearização no ponto a temos:

$$\delta(f(a^*), f(a)) \simeq \frac{|f(a) + (\nabla_a f) \cdot (a^* - a) - f(a)|}{|f(a)|} = \frac{\delta \cdot \|a\| \cdot |(\nabla_a f) \cdot u|}{|f(a)|}$$

Assim tirando os termos quadráticos em δ, o máximo erro que podemos cometer é:

$$\begin{array}{c} \delta_2(a^*, a) = \delta \Rightarrow \delta(f(a^*), f(a)) \leq \delta \cdot \dfrac{\|a\|_2 \cdot \|\nabla_a f\|_2}{|f(a)|} \\ \delta_1(a^*, a) = \delta \Rightarrow \delta(f(a^*), f(a)) \leq \delta \cdot \dfrac{\|a\|_1 \cdot \|\nabla_a f\|_\infty}{|f(a)|} \\ \delta_\infty(a^*, a) = \delta \Rightarrow \delta(f(a^*), f(a)) \leq \delta \cdot \dfrac{\|a\|_\infty \cdot \|\nabla_a f\|_1}{|f(a)|} \end{array}$$

(estas desigualdades são salvo termos em δ^2)

Para qualquer função $f(x)$ diferenciável num ponto a, para qualquer valor $p \in [1, +\infty]$ e para o valor complementar (aquele q com $\frac{1}{p} + \frac{1}{q} = 1$) temos:

$$\begin{array}{c} \Delta(f(a^*), f(a)) \leq \|\nabla_a f\|_q \cdot \Delta_p(a^*, a) + o(\Delta_p(a^*, a)) \\ \delta(f(a^*), f(a)) \leq \dfrac{\|a\|_p \cdot \|\nabla_a f\|_q}{|f(a)|} \cdot \delta_p(a^*, a) + o(\delta_p(a^*, a)) \end{array}$$

Estas fórmulas são chamadas as **fórmulas fundamentais da propagação de erros**

Os números de condição estudados foram obtidos a partir duma linearização. Assim temos desigualdades no limite, quando nos aproximamos dum ponto a, e válidas salvo termos quadráticos.

Vamos tratar de dar uma fórmula que permita majorar o erro de maneira real, sem deixar de lado termos quadráticos, e portanto sem termos que ficar próximos dum ponto a.

Consideremos dois pontos a, a^* de $\mathbb{R}^n$ quaisquer. Consideremos um conjunto $I \subseteq \mathbb{R}^n$, produto de intervalos e que contém a, a^*, e de maneira que as derivadas parciais de f existem não só no ponto a senão em todos os pontos $x \in I$. Então o teorema de Lagrange permite provar o seguinte resultado:

Proposição 1.39 (Fórmula fundamental da propagação do erro absoluto). *Seja $I \subset \mathbb{R}^n$ um produto de intervalos, seja $f(x)$ uma função definida em I, onde sabemos que existem*

as derivadas parciais $\partial_i f(x)$ em cada ponto $x \in I$. Se a, a^* ão pontos de I, existem pontos $p_i \in I$ onde:

$$\begin{aligned} f(a^*) - f(a) = &\partial_1 f(p_1) \cdot (a_1^* - a_1) + \\ &\partial_2 f(p_2) \cdot (a_2^* - a_2) + \\ &\partial_3 f(p_3) \cdot (a_3^* - a_3) + \\ &\dots \\ &\partial_n f(p_n) \cdot (a_n^* - a_n) \end{aligned}$$

Em particular se existe um valor M que serve como limite superior de todos os valores $|\partial_i f(p)|$ em todos os pontos $p \in I$ indicados, tem-se:

$$\Delta(f(a^*), f(a)) \leq M \cdot (|a_1^* - a_1| + |a_2^* - a_2| + \ldots + |a_n^* - a_n|)$$

Observe que a soma indicada representa $\Delta_1(a^*, a) = \|a^* - a\|_1$, e se chamamos $\|\nabla f\|_I$ o valor supremo dos possíveis valores $\|\partial_i f(p)\|$ nos pontos $p \in I$ temos:

$$\boxed{\Delta(f(a^*) - f(a)) \leq \|\nabla f\|_I \cdot \Delta_1(a^*, a)}$$

Demonstração. Basta escrever:

$$\begin{aligned} f(a^*) - f(a) = &f(a_1^*, a_2, \dots, a_n) - f(a_1, a_2, \dots, a_n) + \\ &f(a_1^*, a_2^*, a_3, \dots, a_n) - f(a_1^*, a_2, \dots, a_n) + \\ &f(a_1^*, a_2^*, a_3^*, \dots, a_n) - f(a_1^*, a_2^*, a_3, \dots, a_n) + \\ &\dots + \\ &f(a_1^*, a_2^*, \dots, a_n^*) - f(a_1^*, a_2^*, \dots, a_{n-1}^*, a_n) \end{aligned}$$

e aplicar em cada uma destas diferenças o teorema do valor médio de Lagrange, para a função $h(t) = f(a_1^*, \dots, t, \dots, a_n)$ no intervalo de valores t limitado por a_i^* e por a_i. □

A fórmula indicada proporciona um limite superior para os erros absolutos cometidos. Uma desvantagem é que para ser aplicada a fórmula seria necessário encontrar um majorante das derivadas parciais e no fim temos um erro não é "preciso", sendo possível que o erro esteja limitado por valores ainda menores do que o indicado. Uma vantagem é que esta fórmula não faz intervir termos do tipo $o(\epsilon)$,

Quando trabalhamos com arredondamentos a^* com erros pequenos respeito dum valor a, aplicar f supõe obter valores $f(a^*)$ que, vistos como arredondamento de $f(a)$, levam um erro proporcional ao erro original. A constante de proporcionalidade é chamada número de condição.

Definição 1.40. *Chamamos **número de condição duma função f para o erro absoluto**, medido com respeito duma norma $\|\cdot\|$ o valor:*

$$\lim_{d \to 0^+} \sup_{\Delta(a^*, a) = d} \frac{\Delta(f(a^*), f(a))}{\Delta(a^*, a)}$$

*Chamamos **número de condição duma função f para o erro relativo**, medido com respeito duma norma $\|\cdot\|$ o valor:*

$$\lim_{\delta \to 0^+} \sup_{\delta(a^*, a) = \delta} \frac{\delta(f(a^*), f(a))}{\delta(a^*, a)}$$

Se a função for diferenciável no ponto a e usamos a norma-p, estes números de condição são, respetivamente, os valores

$$K_p(a) = \|\nabla_a f\|_q$$
$$\kappa_p(a) = \frac{\|a\|_p \cdot \|\nabla_a f\|_q}{|f(a)|}$$

Quando se tem a^* um valor aproximado de a com erro relativo δ, a solução proporcionada por f pode ter um erro relativo de até $\kappa \cdot \delta$. A resolução do problema produz resultados onde o erro relativo vê-se multiplicado por κ respeito do erro dos dados introduzidos.

Se $\kappa = 8$, valores de entrada x^* com precisão de p algarismos binários significativos produzem resultados de saída $f(x)$ que podem ter até 8 vezes o erro relativo original, portanto com uma precisão de menos 3 algarismos binários significativos (porque o erro relativo poderia ficar multiplicado com $8 = 2^3$)

A função $f(x)$ diz-se **mal condicionada** num ponto a se o número de condição neste ponto for grande.

O número de condição também pode ser estudado componente a componente (de maneira similar às derivadas parciais). Se trabalhamos com f num ponto a e consideramos a componente a_i como variável, temos uma função $f(a_1, \ldots, t, \ldots, a_n)$ numa única variável t que irá ter um número de condição no ponto a_i dado por:

$$\kappa_{[i]}(a) = \frac{|a_i| \cdot |\partial_i f(a)|}{|f(a)|}$$

Segundo este número de condição, salvo termos quadráticos em $|a_i^* - a_i|$, os erros relativos em cada componente levam à fórmula:

$$\delta(a_i^*, a_i) \leq \delta_i \Rightarrow \delta(f(a^*), f(a)) \leq \delta_i \cdot \kappa_{[i]}(a)$$

Para problemas mal condicionados não importa o algoritmo programado, devemos esperar a possibilidade de que com dados de entrada muito precisos possamos recuperar valores de saída muito imprecisos.

Exemplo

Consideremos a função $f(b,c) = \frac{-b-\sqrt{b^2-4c}}{2}$, fórmula resolvente que determina a menor das raízes no polinómio $x^2 + b \cdot x + d$.

Quando introduzimos os valores (b,c) num computador, se usamos a norma-∞ sabemos que o par (b,c) é introduzido na forma $(b^*, c^*) = (fl(b), fl(c))$, com um erro relativo não superior a 2^{-p} (onde p é a precisão). Podemos esperar que $f(b^*, c^*)$ seja uma boa aproximação da raiz $f(b,c)$? Depende de qual é o número de condição relativo (com a norma ∞) da função f no ponto (b,c).

$$\frac{\partial f}{\partial b}(b,c) = \frac{-b-\sqrt{b^2-4c}}{2\sqrt{b^2-4c}}, \qquad \frac{\partial f}{\partial c}(b,c) = \frac{1}{\sqrt{b^2-4c}}$$

$$\|\nabla_{(b,c)} f\|_1 = \left|\frac{-b-\sqrt{b^2-4c}}{2\sqrt{b^2-4c}}\right| + \left|\frac{1}{\sqrt{b^2-4c}}\right| = \frac{2+b+\sqrt{b^2-4c}}{2\sqrt{b^2-4c}}$$

(nesta última igualdade assumimos $b > 0$ para poder retirar o valor absoluto, dado que $b > \sqrt{b^2-4c} > 0$)

Nesta situação temos como número de condição com a norma-∞ o seguinte:

$$\kappa_\infty(b,c) = \frac{2(2+b+\sqrt{b^2-4c})}{2\sqrt{b^2-4c}(b+\sqrt{b^2-4c})} \cdot \max(|b|,|c|) = \frac{2+b+\sqrt{b^2-4c}}{b^2-4c+b\sqrt{b^2-4c}} \cdot \max(|b|,|c|)$$

Exemplo

Pensemos no caso particular da procura da maior raiz no polinómio $x^2/2 + a \cdot x + 5 = 0$, dependente do parâmetro $a \in \mathbb{R}$, e cuja solução está dada pela função $f(a) = -a + \sqrt{a^2 - 10}$. Sempre que a esteja um pouco longe da $\sqrt{10}$ (digamos $10/a^2$ pequeno), tem valores de κ pequenos:

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

Para estes valores de a, o nosso é um problema bem condicionado. Nas cercanias de $\sqrt{10}$*, no entanto, o problema está mal condicionado. Calcular a raiz mais elevada do polinómio* $x^2/2 + a \cdot x + 5 = 0$*, para valores* a *próximos de* $\sqrt{10}$ *pode levar a resultados muito diferentes, com uma pequena alteração de* a*. Assim, para* $a = 3.16229$ *a raiz do polinómio é aproximadamente* -3.1534557*, para* $a = 3.16228$ *a raiz do polinómio é aproximadamente* -3.158433 *(observamos que só foi alterado o sexto algarismo significativo, e o valor do resultado tem um erro no terceiro algarismo significativo), e para* $a = 3.16227$*, nem sequer existem raízes reais deste polinómio.*

Análise de erros do algoritmo

Temos estudado a parte do erro $\Delta(f(a^*), f(a))$, devida ao arredondamento de a através de a^*. Este é um erro devido à natureza do problema, e não tem nada a ver com o algoritmo utilizado na sua resolução numérica. Há outra componente de erro devida ao algoritmo numérico: $\Delta(f^*(a^*), f(a^*))$.

De maneira imprecisa diremos que um algoritmo é estável se erros pequenos nos dados introduzidos levam a erros pequenos nos resultados numéricos obtidos através do algoritmo.

O algoritmo não é uma função definida sobre todos os números reais, senão só sobre aqueles representáveis na máquina. Portanto, uma caracterização mais precisa desta definição exige um estudo diferente do que o feito para o condicionado do problema.

Esta componente $\Delta(f^*(a^*), f(a^*))$ é uma fonte de erros e podemos pretender limitar os mesmos. A análise do erro introduzido pelo algoritmo pode ser feito desde dois pontos de vista: a análise de erros direta e a análise de erros inversa.

Análise de erros direta

A análise de erros direta dum algoritmo é similar a análise já feita. é o estudo de qual é o erro (absoluto ou relativo) cometido ao tomarmos $f^*(a^*)$ como aproximação de $f(a^*)$:

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

Um algoritmo será tanto melhor se este valor é sempre menor do que a unidade de arredondamento u. De facto, se para todo o a^* representável na máquina o algoritmo satisfaz:

$$f^*(a^*) = f(a^*) \cdot (1+\delta), \qquad \delta < u$$

então a substituição de f pelo algoritmo f^* não produz erro nenhum que não fosse já um erro intrínseco do sistema de arredondamento em ponto flutuante (o sistema não sabe distinguir $f^*(a^*)$ de $f(a^*)$ se $\delta(f^*(a^*), f(a^*)) < u$). Podemos assumir que f^* produz como resultado precisamente o mesmo do que f. Se $\delta < u$ ou δ é duma ordem próxima a u, diz-se que o algoritmo é estável (desde o ponto de vista da análise de erros direta).

Análise de erros inversa

Nesta perspetiva, a pergunta que fazemos não é se $f^*(a^*)$ é aproximadamente igual do que $f(a^*)$ para o sistema de arredondamento da máquina, senão que nos perguntamos se $f^*(a^*)$ é exatamente $f(a)$ para algum a que a máquina percebe como sendo aproximadamente igual do que a^* (isto é, um a tal que $a = a^* \cdot (1+\delta)$ com $\delta < u$).

Exemplo

Suponhamos que temos o polinómio dependendo de a *já estudado* $x^2/2 + a \cdot x + 5$ *e que procuramos, para cada valor de* a*, qual é a raiz maior do polinómio. Suponhamos que a resposta é obtida, para cada* a^**, através dum algoritmo que devolve:*

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

(este é o polinómio de Taylor da função $f(a)$ centrado em $11/2$ e truncado no terceiro passo)

Se queremos estudar o erro cometido pelo algoritmo em $a^* = 5.4$, o método direto diz-nos:

$$\delta = \delta(f^*(5.4), f(5.4)) = 1.08796 \cdot 10^{-3}$$

O método inverso exige computar um valor a com $f^*(5.4) = f(a)$, que é $f^*(5.4) = f(5.40476802)$. O erro relativo segundo esta perspetiva é

$$\delta(a^*, a) = \delta(5.4, 5.40476802) = 8.82967 \cdot 10^{-4}$$

A análise de erros inversa desenvolveu-se com posterioridade à análise de erros direta e permitiu determinar como, ainda que certos algoritmos pareciam fracos quando se estudava o seu erro pelo método direto, o estudo com o método inverso permite justifica-los.

Se o valor de δ com o método inverso é menor do que a unidade de arredondamento u, podemos afirmar que o nosso algoritmo não produz nenhum erro que não fosse já um erro intrínseco do sistema de arredondamento dos dados em ponto flutuante (o sistema não sabe distinguir a^* de a se $\delta(a^*, a) < u$). Podemos assumir que f^* produz a solução exata. Se $\delta < u$ ou δ é duma ordem próxima a u, diz-se que o algoritmo é estável (desde o ponto de vista da análise de erros inversa).

Definição 1.41. *Um **algoritmo** f^* que aproxima uma função f diz-se que é **estável em** a^* **para análise de erros direta** se:*

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

para uma constante c_1 não demasiado grande.

*Um **algoritmo** f^* que aproxima uma função f diz-se que é **estável em** a^* **para análise de erros inversa** se existe um a com $f(a) = f^*(a^*)$ e tal que:*

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

para uma constante c_2 não demasiado grande.

Se $fl(x)$ representa o valor do sistema numérico que aproxima x com menor erro, o ideal no cálculo de f seria um algoritmo que produz sempre como resposta $f^*(a^*) = fl(f(a^*))$

Erros das operações aritméticas

Pensemos nas operações aritméticas elementares, como são a soma e o produto. Assumimos que conseguimos programar um algoritmo de soma estável, por exemplo $a^* \oplus b^* = fl(a^* + b^*)$. Isto resolve a questão do erro que o algoritmo introduz, muito próximo da unidade de arredondamente. Ainda temos a questão de se a soma $s(a, b) = a + b$ é uma função bem condicionada. Para esta função $s(x, y) = x + y$ temos $\nabla_{(a,b)} s = (1, 1)$, vetor com norma-1 simples de calcular: $\|(1, 1)\|_1 = 2$. Portanto o número de condição para erros relativos (com a norma infinito) é:

$$\kappa_\infty(a, b) = \frac{\|(a, b)\|_\infty \cdot 2}{|a + b|}$$

este número de condição é elevado se $a + b$ for uma ordem de grandeza muito inferior aos valores $|a|$, $|b|$. Por exemplo, em $(a, b) = (1001, -1000.9)$ temos número de condição associado $1001/0.1 = 10010$

A operação de soma está mal condicionada quando é feita com valores de soma próxima a 0 (onde "próxima" é em comparação à ordem de grandeza dos elementos que são somados). Dizemos que executar uma soma de elementos quase opostos (ou a diferença de dois números

próximos) é uma operação com **cancelamento catastrófico**, no sentido que pequenos erros nos dados de entrada implicam grandes erros no resultado da operação.

Para o produto podemos fazer o mesmo tipo de análise. No entanto vamos estudar a propagação do erro com ajuda do número de condição em cada componente. Se temos $f(x,y) = x \cdot y$, sabemos $\partial_x f(a,b) = b$, $\partial_y f(a,b) = a$. Portanto:

$$\kappa_x(a,b) = \frac{|a| \cdot |b|}{|a \cdot b|} = 1, \qquad \kappa_y(a,b) = \frac{|a| \cdot |b|}{|a \cdot b|} = 1$$

e para erros relativos podemos afirmar:

$$\delta(a^*,a) \leq \delta_x \wedge \delta(y^*,y) \leq \delta_y \Rightarrow \delta(a^*b^*,ab) = \delta(f(a^*,b^*),f(a,b)) \leq \kappa_x(a,b)\cdot\delta_x + \kappa_y(a,b)\cdot\delta_y = \delta_x + \delta_y$$

se as componentes não trazem um erro relativo grande, o produto não irá ter um erro relativo grande.