Métodos Numéricos

Um curso para o Mestrado Integrado em Engenharia Informática e Computadores da FEUP

Carlos Madureira
Cristina Vila
José Soeiro de Carvalho

Faculdade de Engenharia da Universidade do Porto
Departamento de Engenharia de Minas

Este documento é uma versão provisória, de utilização exclusiva no âmbito da disciplina de Métodos Numéricos do Mestrado Integrado em Engenharia Informática e Computadores da Faculdade de Engenharia da Universidade do Porto.

Todas as correcções e contribuições são bem-vindas!

Carlos M. N. Madureira (cmad@fe.up.pt)
Maria Cristina C. Vila (mvila@fe.up.pt)
José M. S. Soeiro de Carvalho (jmsoeiro@fe.up.pt)

Conteúdo

Lista de Figuras

Lista de Tabelas

Introdução

Na prática da computação digital científica verificam-se numerosos e diversificados tipos de situações relacionadas com problemas de Análise Numérica. As que se revelam mais simples são aquelas em que se pretende construir um algoritmo para utilizar apenas uma vez, ou, quando muito, um número muito reduzido de vezes, para resolver um dado problema concreto e em que se tem uma ideia razoável do modo como vão desenvolver-se os cálculos. Nestes casos os problemas de *tempo de processamento* não se põem (pelo menos de modo agudo, sobretudo quando dispomos de um computador pessoal que podemos deixar a trabalhar durante a noite só para nós) e *as questões de precisão* são relativamente simples de tratar (até porque se dispõe, o deve dispor, *a priori* de ordens de grandeza para as soluções, o que permite controlar os erros mais óbvios); porém, a tentação de criar soluções extremamente dependentes das características particulares do problema e das peculiaridades da máquina é quase irresistível, de tal modo que os programas desenvolvidos não têm, em geral, qualquer transportabilidade.

Um segundo tipo de situação é o de um algoritmo destinado a correr diversas vezes em diversos tipos de situações todas incluídas em um mesmo contexto mas destinadas em princípio a um único utilizador, e difere da anterior no facto de existir uma potencial diversidade de dados de partida conducente a diferentes tipos de problemas numéricos. A principal dificuldade operacional reside, neste caso, no esforço de imaginação necessário para prever as diversas classes de situações que virão eventualmente a verificar-se, sob pena de se ter que rescrever sucessivas versões do mesmo algoritmo, uma para cada falhanço detectado. A situação torna-se nitidamente mais complicada quando o algoritmo se destina a ser corrido por diversos utilizadores, porque, nesse caso, é virtualmente impossível prever *a priori* o tipo de complicações numéricas que, em função de diferentes dados de partida, podem vir a ocorrer. Este é o caso típico dos *programas de biblioteca de uso geral*. Neste caso torna-se necessário usar dos maiores cuidados para cobrir tantas situações diferentes quantas seja possível, tendo ao mesmo tempo a habilidade de não produzir programas grosseiramente ineficientes ou tão complicados que desencorajem o utilizador comum. A real dificuldade de conciliar objectivos tão contraditórios e a subjectividade inevitável na ponderação das diferentes prioridades levam muitos utilizadores (incluindo os autores deste curso) a olhar com alguma desconfiança a tendência de certos outros para o uso sistemático de rotinas de biblioteca; o preço de uma tal atitude é, inevitavelmente, a necessidade de escrever os seus próprios algoritmos e programas.

Uma terceira classe de situações corresponde aos problemas computacionais tão grandes, tão complexos, tão particulares ou tão novos que se torna necessário, para os resolver, um conhecimento muito profundo da questão concreta que os gerou e o recurso à intuição física do proponente, que se supõe ser o seu melhor conhecedor. Esta classe de problemas dá, normalmente, origem a soluções particulares, que usam técnicas adequadas às peculiaridades de cada problema, ou de cada formulação particular do problema. O uso dessas soluções por outras pessoas que não o seu próprio autor exige um estudo muito aprofundado que pode, frequentemente, revelar-

se infrutífero e/ou impraticável, dada a conhecida tendência dos programadores, sobretudo dos mais criativos, para documentarem mal os seus produtos.

Finalmente, uma outra classe de situações ocorre nas aplicações em tempo real (como os programas de controlo de um processo industrial); nestes casos, o imperativo fundamental, para além de uma precisão (que, ao contrário do que poderia pensar-se, nem sempre será crítica) será o de uma execução extremamente rápida; em casos deste tipo, a rapidez da execução pode mesmo compensar em larga medida uma certa imprecisão dos resultados, desde que não exagerada. Tal como no caso anterior, deve fazer-se um esforço importante para garantir que se dispõe de toda a informação relevante sobre o problema e sobre os seus possíveis métodos de resolução, bem como de todo o tempo necessário para desenvolver uma solução realmente eficaz. Estes dois últimos casos constituem o domínio de eleição da aplicação das habilidades e artimanhas próprias da arte do analista e do programador.

Em resumo, existem três classes fundamentais de necessidades de algoritmos e programas computacionais:

- soluções expeditas, facilmente disponíveis ou implementáveis, embora possivelmente pouco eficientes, mas não imprecisas;
- soluções cuidadosamente pensadas, precisas, rápidas, seguras e de uso geral;
- soluções particulares para problemas particulares, em que as peculiaridades da aplicação em vista podem desempenhar papel central.

Como é óbvio, o presente curso só trata dos dois primeiros tipos de problemas. O nosso ponto de vista central será, portanto, o de que, embora o objectivo primário seja, como sempre, o de sacar resultados da máquina, esses resultados se destinam a fundamentar decisões, por vezes de importância vital, pelo que se torna necessário, antes de mais, compreender claramente e em profundidade o que tais números podem dizer, ou não dizer, o que podem valer, ou não valer. Assim, considerando que se trata de cadeira de mera introdução colocada muito cedo no plano do curso, a selecção de matérias tem em vista aquilo que de mais básico se prende com os objectivos centrais do exercício da profissão de engenheiro a nível superior, e não aquilo que está mais em voga ou mais bem documentado na literatura técnica e científica; secundariamente, a selecção das matérias nem pretende cobrir todas as necessidades que possam surgir no exercício da profissão, nem escolher os "melhores"métodos no sentido do uso eficiente dos recursos das máquinas - que, ainda por cima, mudam quase todos os dias -, mas apenas seleccionar aquelas matérias que melhor possam compensar o tempo e o esforço gastos no seu estudo.

Uma tal selecção é notavelmente dificultada

- pelo facto de *não existir, por trás da análise numérica, um corpo de doutrina coerente*, o que faz com que as soluções numéricas realmente eficientes e adaptadas sejam frequentemente produtos artesanais frutos de processos criativos baseados em toda uma experiência anterior do autor-utilizador;
- *pela enorme diversidade de métodos, técnicas, variantes e variações concebidas até ao presente* e que a maior parte dos tratadistas tende a apresentar a granel, sem qualquer senso crítico;

- pelo facto de muitas dessas técnicas serem *meros artifícios sem interesse geral*, facto que não pode ser revelado senão por um uso intensivo e constantemente crítico que o ensino convencional não encoraja.

A selecção de métodos aqui apresentada representa, portanto, um mero ponto de vista muito pessoal do autor, caucionado apenas pela experiência investigacional e docente da equipa que tem dirigido no Departamento de Engenharia de Minas ao longo das duas últimas décadas.

Por outro lado, o espaço de escolha encontra-se amputado pelo facto de existir no curriculum uma cadeira de Investigação Operacional, no âmbito da qual recaem os métodos cuja fundamentação é de carácter essencialmente algébrico - programação linear, algoritmos de árvores e redes, etc. Neste contexto, o capítulo sobre sistemas de equações lineares aparece como u, tanto estranho no curso, dado que a sua metodologia releva dos métodos algébricos.

A construção do curso no seu conjunto assenta sobre uma opção táctica muito clara, resultante de muitos anos de ensino da matéria ao nível da licenciatura: *total separação entre a Análise Numérica* - entendida como construção de algoritmos e análise do seu comportamento face à inevitável finitude da representação concreta dos números dentro da máquina - e *a programação desses algoritmos* no contexto de uma linguagem e/ou de uma máquina particular. Com efeito, toda a experiência passada nos mostrou para além de qualquer dúvida razoável que a mistura imprudente das duas abordagens e dos respectivos pontos de vista e polarizações faz com que o estudante médio, engodado pela preocupação, não de todo ilegítima mas limitativa, da produtividade imediata, concentre a totalidade do seu esforço no segundo e esqueça por completo o primeiro, aquele que constitui, precisamente, o objectivo primordial da cadeira; acontece que o ponto de vista peculiar da programação valoriza desproporcionadamente os problemas da lógica e da economia do algoritmo e tende a remeter para segundo plano os problemas vitais da precisão, problemas que são de carácter muito menos racionalizável e, por esse facto, de abordagem muito mais árdua para o estudante de formação racionalista.

Um outro ponto de vista didáctico que confere um certo grau de originalidade à estrutura da cadeira é o de se praticar sistematicamente um certo tipo de "reconstrução"da história dos métodos da Análise Numérica com a intenção de mostrar como os métodos se encadeiam logicamente uns com os outros e, nessa perspectiva, podem constantemente ser redesenhados, o que favorece uma das atitudes mentais mais importantes para o candidato a analista numérico.

A necessidade do uso, na cadeira, de algoritmos implementados em computador, indispensável para a aquisição do seu completo domínio pelo estudante, cria um dilema extremamente difícil de ultrapassar: por um lado, os cuidados necessários em matéria de precisão são pouco compatíveis com programação de principiantes e, por outro lado, como se depreende do que ficou dito no início, temos sérias reservas quanto ao uso cegos de programas de biblioteca. Por isso, todos os exemplos e manipulações exibidas e todos os exercícios propostos se destinam a ser trabalhados à mão ou à máquina de calcular ou, na melhor das hipóteses, em folha de calculo; uma tal estratégia, quando bem praticada, permite seguir passo a passo eventuais incidentes resultantes de problemas de representação; no entanto, uma vez dominados os princípios e os conceitos, deve ser feito um esforço sério para formalizar a aprendizagem sob a forma de construção de programas informáticos que incorporem explicitamente todos os critérios de segurança e precaução que a teoria e a prática "à mão"aconselharam.

Finalmente, na concepção e implementação do curso, um outro problema se levantou: o da

recente vulgarização dos *manipuladores simbólicos*, ferramentas informáticas que,

- ao contrário das vulgares linguagens de programação, têm a possibilidade de manipular directamente expressões simbólicas e não apenas números;
- permitem representar os números (dados, resultados intermédios e finais) com precisão "infinita"(dentro, naturalmente, dos limites de memória da máquina), representando os racionais sob a forma de quocientes de inteiros de comprimento adequado e mantendo em suspenso todas as operações irracionais; em opção, um modo de cálculo aproximado permite trabalhar com um número arbitrário de algarismos significativos.

Nestas condições, os manipuladores simbólicos permitem a verificação e o controlo do cálculo programado em linguagens convencionais e é um dos objectivos do curso levar o estudante a encará-los como tal.

Estamos mesmo em crer que, dentro da evolução previsível das capacidades de memória e das velocidades de cálculo dos computadores das próximas gerações, esta é uma das direcções genéricas em que se desenvolverá o cálculo científico nas próximas décadas, o que tornará obsoletas muitas das abordagens tradicionais da análise numérica. Pensamos, com efeito, que a generalização do uso dos manipuladores simbólicos corre o risco, se não for devidamente pensada, estudada e integrada, de vir a desempenhar, em relação ao ensino superior, o mesmo papel ambíguo, e por isso confusionista e dissolvente, que a generalização do uso das calculadoras de bolso desempenhou, em relação ao secundário, nas décadas de 70 e 80. Por outro lado, há que reconhecer que tais ferramentas informáticas se encontram apenas na fase de utilitários para a realização de cálculos avulsos e não podem ainda ser eficientemente integradas em programas construídos pelo utilizador; isso impede, naturalmente, a completa centragem nelas de uma cadeira que não pode dar-se ao luxo de ignorar as reais necessidades da actual computação científica e técnica. Assim, incapaz, neste momento, de desenvolver um curso completo ao longo desta linha estratégica, o autor limitou-se a fazer apenas algumas tímidas tentativas de introdução ao uso de um manipulador simbólico muito acessível e que bem conhece (DERIVE, A Mathematical Assistant ^{TM}, The Software House, Inc.) a propósito de alguns problemas mais delicados levantados pelo desenvolvimento de um curso que é ainda fundamentalmente convencional.

[DB74]
[CB81]
[Gol91]
[Ham71]
[Lie68]
[Moo63]
[Mul89]
[Wal90]

1 O Erro em Análise Numérica

Contents

Figures

Tables

1.1 Matemáticas e Análise Numérica

> O senhor I. N. Génuo pretende fazer uma aplicação bancária de longo prazo para garantir o futuro da sua família; para isso, procura o Banco Caótico Português, que lhe oferece o seguinte plano-poupança:
>
> > você faz uma entrega inicial de $(e-1)$ euros (sendo $e = 2.718281828...$) e ao fim de 1 ano multiplicamos o seu capital por 1 e subtraímos 1 euro para despesas administrativas; ao fim de 2 anos multiplicamos o seu capital por 2 e subtraímos 1 euro para despesas administrativas; ao fim de 3 anos multiplicamos o seu capital por 3 e subtraímos 1 euro para despesas administrativas; e assim sucessivamente, até que ao fim de 25 anos você pode retirar todo o dinheiro acumulado.
>
> Cuidadoso, o senhor I. N. Génuo não se compromete e vai para casa pensar. Chegado a casa pega na sua Casio FX702P para saber quanto terá ao fim de 25 anos e o resultado aterroriza-o: o saldo será devedor de `140 302 545 600 000` euros! Duvidando da sua competência de calculador, corre a um amigo informático que tem uma estação Sun Sparc e pede-lhe que calcule o saldo. A resposta é positiva: `4 645 987 753` euros! Satisfeito, corre ao Banco para subscrever esse interessante plano. Ao fim de 25 anos tem um desgosto: o Banco entrega-lhe `4` milésimos de cêntimo. Este é um exemplo típico de cálculo instável que, devido a erros de arredondamento nos dados iniciais e nos resultados intermédios, pode arrastar enormes variações no resultado final.
>
> J. M. Muller, Ordinateurs en quête d'arithmétique
>
> La Recherche, N^o Spécial, Juillet/Août 1995, adaptado

As *Matemáticas convencionais* (sobretudo quando consideradas no sentido em que são correntemente ensinadas nos nossos cursos superiores) diferem significativamente da *Análise Numérica*, na realidade muito mais significativamente que o que habitualmente se considera. A diferença mais evidente reside no uso constante da noção de *infinito* que nelas se faz,

- tanto a propósito da representação dos números, que são sempre considerados como representados com precisão infinita;
- como a propósito de processos de cálculo (passagem ao limite, cálculo de uma derivada, soma de uma série, cálculo de um integral definido),

enquanto que a computação, que é o objecto da Análise Numérica, é sempre feita por meio de dispositivos, manuais ou automáticos, dotados de precisão inelutavelmente finita e que operam em tempo finito (e, se possível, muito curto!).

Exemplo 1.1 Matemática, números e processos

Números típicos das matemáticas:

$$\pi = 3.14159265358979323846264338327950288419716939937510\ldots$$

$$1/3 = 0.33333333333333333333333333333333333333333333333\ldots$$

Processos típicos das matemáticas:

$$\frac{dy}{dt} = \lim_{\Delta x \to 0} \frac{\Delta y}{\Delta x}$$

$$\int_a^b f(x)\, dx = \lim_{\Delta x_i \to 0} \sum_{i=1}^{\frac{b-a}{\Delta x_i}} f(x_i)\Delta x_i$$

Mesmo erros de arredondamento muito pequenos, quando acumulados ao longo de processos de cálculo complexos, podem ter efeitos extremamente perversos, dando origem a perdas de precisão inesperadas, sem qualquer comum medida com as causas que lhes deram origem. Este facto, nem sempre devidamente apreciado pelos principiantes, é, no entanto, de importância fundamental, e a falta do seu reconhecimento expresso é causa frequente de muitas confusões e de graves acidentes; no dizer lapidar de [Ham71]

> "só a parte das matemáticas convencionais que se revela resistente a estes efeitos perversos tem interesse para a análise numérica, assim como para qualquer outra aplicação das matemáticas ao mundo real".

Só por este facto, a Análise Numérica seria, de direito próprio, parte fundamental das Matemáticas, no que se refere à sua aplicabilidade, embora constituindo o que temos que reconhecer como um ramo não-convencional destas últimas.

Outras diferenças entre as Matemáticas convencionais e a Análise Numérica são mais subtis ou mesmo mais encapotadas, por resultarem do uso em sentidos profundamente diferentes dos mesmos termos e expressões. Assim, por exemplo, como veremos muito em breve, a expressão *zero de uma função* pode ter diferentes sentidos em análise numérica, nenhum deles coincidente com o que tem em matemática.

Outras diferenças ainda são as que residem em profundas divergências de objectivos, de gostos e mesmo de conceitos de elegância entre as Matemáticas convencionais e a Análise Numérica. Assim, por exemplo, os teoremas de existência, tão caros aos matemáticos, raramente têm qualquer interesse em análise numérica; do mesmo modo, a Matemática prefere teoremas exactos e precisos, demonstráveis e demonstrados, enquanto a Análise Numérica frequentemente se satisfaz com resultados meramente plausíveis e com métodos heurísticos, intuitivos.

As consequências destas diferenças, nem sempre óbvias nem apreciadas, são, no entanto, sempre importantes e, em certos casos, podem prestar-se a confusões perigosas. Assim, por exemplo, enquanto em certos casos o resultado de um cálculo numérico pode considerar-se como uma aproximação satisfatória ao ideal matemático, em numerosos outros casos a diferença é tal que nem sequer se pode considerar o resultado como minimamente aproximado; no primeiro caso, a tendência corrente é para, sem mais análise, imputar a diferença aos "defeitos"inerentes à análise numérica; no segundo caso, por extensão, tende a pensar-se que se trata simplesmente de uma variante exagerada do caso anterior, isto é, a culpa é automaticamente atribuída ao cálculo numérico; ora, em muitos casos, se não mesmo na maioria, é precisamente o modelo matemático, por exacto que pareça, que está em falta, por se não adequar minimamente à realidade em estudo, e

esta realidade tem uma dupla face: a do problema físico subjacente ao modelo matemático e a do sistema computacional subjacente à resolução do modelo matemático.

1.2 A representação dos números numa máquina

> **Vamos explorar agora a dicotomia numérico versus digital! ...** citando (apocrifamente ?) um conhecido compositor (1981).

Para além do facto importante e bem conhecido, e sobre o qual não insistiremos especialmente, de a representação interna dos números nas máquinas electrónicas ser quase sempre *não-decimal*, mas sim binária, octal ou hexadecimal, o analista numérico tem que ter bem presente que os computadores digitais usam duas classes de representações de números que a matemática convencional desconhece em absoluto. Para compreender este facto é preciso, em primeiro lugar, aceitar a ideia de que os números não existem na realidade exterior a nós, de que são apenas construções conceptuais. Assim, por exemplo Conte e de Boor em [CB81] afirmam:

> a Análise Matemática tornou-se uma disciplina exacta apenas quando deixou para trás as restrições do cálculo numérico e passou a tratar exclusivamente os problemas levantados por um modelo abstracto do sistema dos números, o modelo dito dos *números reais*. Este modelo foi formulado intencionalmente para tornar possível uma definição conceptualmente operacionalizável do conceito de limite, o que abre caminho para a resolução matemática de uma multidão enorme de problemas práticos, uma vez que esses problemas sejam traduzidos na linguagem do modelo. No entanto, permanece ainda de pé o problema da retroversão das soluções simbólicas ou abstractas encontradas para a linguagem das soluções práticas. É esta a tarefa assumida pela análise numérica que, ao assumi-la, assume as limitações do cálculo prático, às quais a análise matemática tão elegantemente fugiu. As soluções numéricas são, portanto, apenas tentativas e, na melhor das hipóteses, tentativas precisas apenas dentro de certos limites calculáveis.

Deste modo, na concepção desta classe de autores, a análise numérica não se prende apenas com a construção de métodos numéricos mas, talvez principalmente, com a construção de limites calculáveis para os erros.

Nas matemáticas puras, esse conceito de número não exige representação concreta, individualizada, mas apenas simbólica; pelo contrário, em todos os ramos das matemáticas aplicadas (com os quais a análise numérica tem sempre, directa ou indirectamente, alguma coisa que ver) exige-se uma representação concreta de cada número particular, representação que, para distinguir do conceito abstracto de *número*, designamos por vezes por *numeral*. Em consequência, o número de números representáveis concretamente, embora por vezes indefinido (como sucede no cálculo manual, em que podem utilizar-se tantos algarismos quantos quisermos e em que, portanto, não existe um limite definido para a representação), é sempre necessariamente finito; pelo contrário, nas máquinas esse número é perfeitamente limitado e mesmo, na maior parte dos casos, surpreendentemente pequeno.

Três classes de números são muito importantes para a informática:

inteiros este números são normalmente utilizados nas operações de contagem e enumeração, mas a sua importância na informática digital é enorme; efectivamente o "mundo"de uma máquina digital é discreto e representável por números inteiros;

racionais números que são obtidos pelo quociente entre dois quaisquer números inteiros (excluindo o zero como denominador);

reais estes números são utilizados na medida e na representação de mundos contínuos, de que o nosso parece ser um exemplo[1]; a sua utilização é imposta pelo uso e não pelo suporte.

Exercício 1.1 O leitor tem a certeza de que sabe distinguir *enumeração* de *contagem*? A distinção, embora provavelmente não fundamental em Análise Numérica, é-o certamente em Programação.

A representação de quantidades inteiras não é problemática. Já vimos que é a *forma natural* como os computadores digitais lidam com elas. Mas os números inteiros *informáticos* não correspondem ao conjunto **Z** da matemática: nem o seu domínio é infinito – no caso mais corrente cobrem apenas o intervalo de -32767 a $+32767$, nem as operações sobre eles definidas têm as mesmas propriedades[2]. Usam-se correntemente inteiros construídos sobre palavras de 16, 32 ou 64 bits.

A representação e manipulação de números reais por computadores digitais tem uma característica surpreendente: é impossível!

Com efeito, não podemos representar perfeitamente o contínuo recorrendo ao discreto. A única solução é recorrer a aproximações. No caso dos números reais é recorrer à sua aproximação por racionais.

Um exemplo simples mostra as deficiências dessa aproximação: suponha que o conjunto dos inteiros é constituído apenas pelos seguintes números

$$\{-3,-2,-1,0,1,2,3\}$$

o conjunto de todos os racionais incluiria os inteiros e as fracções

$$\left\{-\frac{3}{2},-\frac{2}{3},-\frac{1}{2},-\frac{1}{3},\frac{1}{3},\frac{1}{2},\frac{2}{3},\frac{3}{2}\right\}$$

o que exclui todas as infinitas fracções que não se podem escrever recorrendo ao inteiros entre -3 e 3, como $\frac{4}{5},\frac{1}{7}$, além de todos os inteiros e fracções mais pequenos que -3 ou maiores que 3.

[1] pelo menos à escala humana...

[2] basta pensar qual será a soma dos dois maiores números do domínio!

1.2.1 Representação, Codificação

A forma de representar internamente a informação está estritamente condicionada pelas características físicas do computador, especialmente por trabalhar com circuitos electrónicos descendentes do transístor, que podem estar em um de dois estados, ligado ou desligado.

Como é registada internamente a informação? Como vimos, o computador trabalha sobre estados eléctricos dos seus componentes internos, normalmente duas voltagens de referência (+5 e -5 Volts, por exemplo). A uma entidade que só pode tomar um deste valores chamamos um bit (de binary digit, ou, porque não, de bit - pedaço, pedacinho). Um bit é também a menor quantidade de informação.

A correspondência entre estes dois estados e os valores lógicos de *verdade* ou *falsidade* é evidente.

No caso dos números poderíamos associar cada um desses dois estados a um dos dígitos binários, 0 e 1. Essa codificação binária não constitui qualquer transtorno para a aritmética...

Por outro lado, um alfabeto apenas com as letras A e B ou uma paleta apenas com branco e preto[3] – cor ou a sua ausência – seriam muito limitativas.

Os problemas que podem ser resolvidos recorrendo ao computador raramente podem ser expressos em termos de bits, sendo mais normal serem expressos em termos de caracteres, números, símbolos e mesmo objectos mais complicados.

000
001
010
011
100
101
110
111

Como utilizar então esta base tão limitada para representar objectos tão diferentes? A solução consiste em recorrer a regras de codificação, utilizando agrupamentos de bits, em vez de bits isolados. Um agrupamento de três bits têm oito estados diferentes, permitindo um alfabeto de oito letras ou uma paleta de oito cores. Uma simples contagem permite verificar que oito estados não são suficientes para conter o alfabeto ocidental (composto de um mínimo de 28 caracteres minúsculos, outros tantos maiúsculos, alguns sinais de pontuação e, já agora, os 10 dígitos da numeração árabe), pelo que se utilizam correntemente agrupamentos de oito bits que podem assumir um de 256 estados diferentes. A cada um desses estados associamos um código, conforme o que se pretende representar seja um número, um dígito, uma letra, uma cor, etc. Os sistemas de códigos não são únicos, existindo no entanto algumas convenções e estandardizações para nos facilitar a vida (uf!). Por exemplo, para a representação de caracteres alfa-numéricos, utilizam-se correntemente dois standards, ambos de origem americana, os códigos ASCII[4] e ANSI[5].

Como se pode ver, todos estes códigos são apenas convenções. O mesmo valor binário pode ser interpretado de diferentes maneiras, sendo essa interpretação da inteira responsabilidade do utilizador.

1.2.2 Representação de quantidades

Para representar quantidades recorremos a símbolos, cujo significado é determinados por regras de codificação, podendo depender da colocação e do contexto. Historicamente, não só os símbolos mudaram, como, mais importante ainda, as técnicas de codificação.

[3] ou já agora laranja e verde!
[4] ASCII - American Standard Code for Information Interchange
[5] ANSI - American National Standards Institute

Voltagem − +	representação binária	representação decimal	Número natural	Número inteiro[6]	Caracter ASCII
−−−−−−−−	00000000	0	0	0	NULL
−−+−−−−+	00100001	33	33	33	!
−+−−−−−+	01000001	65	65	65	B
−+−++−+−	01011010	90	90	90	Z
−++++−+−	01111010	122	122	122	
+−−++−++	10011011	155	155	−27	

Tabela 1.1: Codificação de números e caracteres

São comuns técnicas posicionais e não posicionais de escrita de números. MMCDLXIV

Um exemplo de notação não posicional é a utilizada na numeração romana, em que cada símbolo tem sempre o mesmo valor. Os símbolos utilizados são as iniciais das palavras romanas que designavam as quantidades milhar **M**, meio milhar **D**, centena **C**, meia centena **L**, dezena **X**, meia dezena **V**, unidade **I**. Para calcular a quantidade representada por um número romano é preciso somar todos os dígitos que o compõem, tendo em atenção que eles se escrevem em ordem decrescente da esquerda para a direita, e que se existir uma inversão de valores, um dígito menor à esquerda de um maior, este deve ser subtraído ao valor do seguinte. 2464

Com a técnica de representação ocidental comum, utilizando uma simbologia composta apenas de dez símbolos diferentes (os símbolos 0 1 2 3 4 5 6 7 8 9), somos capazes de representar quantidades conceptualmente ilimitadas[7]. Recorremos para isso a uma técnica de notação posicional, em que o valor associado ao símbolo depende da sua posição no número. No caso dos números inteiros, o dígito mais à direita corresponde às unidades, o imediatamente à esquerda às dezenas, ...

$$(\ldots a_3a_2a_1a_0,a_{-1}a_{-2}\ldots)_b = \ldots + a_3b^3 + a_2b^2 + a_1b^1 + a_0b^0 +, a_{-1}b^{-1} + a_{-2}b^{-2} + \ldots;$$

Normalmente, b é um inteiro maior que 1 e os a são inteiros no intervalo $0 \leq a_k < b$. Os dígitos à esquerda são *mais significativos* que os dígitos à direita[8].

Como avaliar então um número escrito em notação posicional? Notação posicional

O número é escrito como uma sequência ordenada de dígitos, em que, quanto mais à esquerda, maior o valor associado ao dígito.

A cada dígito será atribuído um valor igual ao índice da sua posição na lista de símbolos multiplicada pela base elevada ao índice do dígito no número (sendo positivo o índice dos dígitos da parte inteira e negativo o índice do dígito da parte não inteira), sendo no fim somados todos os valores obtidos (ver exemplo na tabela 1.2).

Este tipo de notação posicional é utilizável qualquer que seja a base de numeração (ou o número de símbolos diferentes que a simbologia nos permitir utilizar). São correntemente utilizadas na informática as bases 2, 8, 10 e 16 .

[7] As limitações são apenas espaciais e temporais.

[8] ver uma descrição detalhada em [**?**].

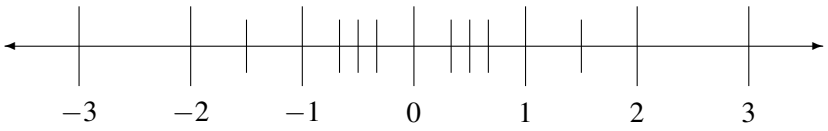


Figura 1.1: Números racionais possíveis para o domínio inteiro [-3,3]

Número na base 16	A8F		
Símbolos	A	8	F
Índice do símbolo	A	8	F
Índice do dígito	2	1	0
Valor do dígito	$A \times 16^2$	8×16^1	$F \times 16^0$
Valor do número	10×16^2	8×16^1	15×16^0
na base 10	2560	128	15
			2703

Tabela 1.2: Cálculo posicional

Base	Designação	Símbolos	101_{10}
2	binário	0 1	1100101
8	octal	0 1 2 3 4 5 6 7	145
10	decimal	0 1 2 3 4 5 6 7 8 9	101
16	hexadecimal	0 1 2 3 4 5 6 7 8 9 A B C D E F	65

Tabela 1.3: Bases Numéricas

Muitas outras bases foram utilizadas historicamente, essencialmente para contagem, umas decalcadas de características humanas, tal como a base dez, dos dez dedos das mãos[9], ou a base três, das três articulações dos dedos, outras com raízes desconhecidas tal como a base vinte usada pelos Maias[10], outras ainda, como a base sessenta da Mesopotâmia, que ainda hoje se utilizam na notação sexagesimal da medida de ângulos ou do tempo.

As mãos são utilizadas muitas vezes na aritmética como um dispositivo auxiliar de contagem, permitindo-nos contar de um a dez (ou recorrendo à famosa IBM do Manelinho[a], contar até vinte). O seu uso como um dispositivo de notação posicional é muito mais eficaz. Consegue imaginar como?

A mão pode ser usada como um dispositivo posicional adoptando regras muito simples: um dedo completamente dobrado corresponde a zero, o dígito menos significativo é o mindinho, o mais significativo é o polegar. Se usarmos uma convenção binária – o dedo todo dobrado vale zero, todo esticado vale um – uma mão permite representar números entre 0 e $2^5 - 1$; se utilizarmos cada uma das articulações dos dedos e do pulso, poderemos usar base 4. Quão enganadora a afirmação "Tenho de meu apenas aquilo que posso contar com uma mão"!
(Radiografia em http://uwmsk.org/RadAnat/HandPALabelled.html)

[a] personagem das tiras desenhadas "Mafalda", de Quino, que calçava sandálias de modo a poder contar também com os dedos dos pés.

1.2.3 Conversão entre bases

Para efectuar uma conversão de bases recorre-se à divisão euclidiana, divisão com resto, divisão inteira ou modular.

se for A a representação do número na base a e B a sua representação na base b então cada um dos dígitos de B será obtido pelo resto da divisão de A por b, em que A é substituído pelo quociente Q da divisão anterior. Um exemplo de aplicação é a conversão de 100_{10} para a base 3 descrita na tabela 1.4.

Iteração	4	← 3	← 2	← 1	← 0
Inicial (base 10)					100
Quociente (base 10)	0	← 1	← 3	← 11	← 33
Resto	1	← 0	← 2	← 0	← 1
Resultado (base 3)					10201

Tabela 1.4: Conversão de 100_{10} para base 3

[9] duvidoso, já que a base adequada seria 5, por permitir agrupamentos de mãos
[10] ou a contagem de javalis em catorzenas na época de Obelix [**?**] – talvez a antecipação do actual $12+2$ grátis.

Repare-se que este algoritmo pode ser expresso quer como uma iteração quer como uma recursão.

1.2.4 Notação

A notação de números não inteiros é feita segundo duas técnicas: em *vírgula fixa*, em que a vírgula separa as unidades das décimas e em *virgula flutuante*, em que o valor dos dígitos à esquerda e à direita da vírgula é determinado por um expoente associado ao número.

Um número representado em vírgula fixa é composto de duas partes: a parte inteira, à esquerda da vírgula, e a parte decimal à direita da vírgula.

Um número representado em vírgula flutuante é composto de três partes: a *parte fraccional* (ou *mantissa*) m, o *expoente* (ou *característica*)e, e a *base* b:

$$mantissa \times base^{expoente}$$

A *mantissa* é por sua vez composta de uma parte inteira e de uma parte decimal.

Duas constatações simples permitem reduzir bastante a complexidade desta representação:

- uma operação simples de normalização consistindo em escrever o número de maneira a que a parte inteira da mantissa seja sempre nula e que todos os dígitos da parte decimal sejam significativos (não existam zeros imediatamente á esquerda da vírgula)

  $$1/b \leq |m| < 1$$

  permite retirar da representação a parte inteira da mantissa e a vírgula;

- a *base* é condicionada pela base do sistema de numeração utilizado, sendo constante dentro do mesmo sistema[11], pelo que a sua escrita é dispensável.

Desta maneira, a representação em vírgula flutuante fica reduzida a duas partes, a parte decimal da mantissa e o expoente, que podem ambas ser representadas como números inteiros (ver tabela 1.5).

Se nos abstrairmos da virgula, um número representado em vírgula fixa também fica reduzido a duas componentes representáveis como inteiros.

fixa	flutuante	
100.10	$= 100.10 \times 10^0$	
	$= 1001 \times 10^{-1}$	
	$= 0.1001 \times 10^3$	normalizada
100 10	$= 1001\ 3$	simplificada

Tabela 1.5: Vírgula fixa e flutuante

Qual é então a notação preferível para o cálculo numérico? O factor Domínio (ver tabela 1.6) faz com que a escolha penda normalmente para a vírgula flutuante.

[11] no sistema de base 10 $1.27 \times 10^2 = 12.7 \times 10^1$ mas $1.27 \times 5^2 \neq 12.7 \times 5^1$.

	Vírgula fixa	Vírgula flutuante
Representação	Parte inteira com N_i e parte decimal com N_d dígitos, $\pm iiii,dddd$	Mantissa com N_m e expoente com N_e dígitos, $\pm 0,mmmm \times b^{\pm eeee}$
Para permitir a comparação, façamos $N = N_i + N_d = N_m + N_e$ (os números ocupam o mesmo espaço) e utilizemos a mesma base B.		
Domínio	depende de N_i, sendo aproximadamente $]-B^{N_i+1}, +B^{N_i+1}[$	depende de N_e, sendo aproximadamente $]-B^{B^{N_e}-1}, +B^{B^{N_e}-1}[$
Granularidade	é constante, valendo B^{-N_d}	é variável, valendo B^{p-N_m-1} (em que p é a potência de dois números consecutivos), podendo por isso tomar valores no intervalo $[B^{-B^{N_e}-N_m-1}, B^{B^{N_e}-N_m-1}]$ com um buraco em torno de zero de $2 \times B^{-N_e}$ o espaçamento relativo entre dois números consecutivos é constante e vale $B^{p-N_m-1}/B^p = B^{-N_m-1}$

Tabela 1.6: Comparação de notações em vírgula fixa e flutuante

A *precisão*, ou *comprimento*, n da mantissa de um número em vírgula flutuante é em geral determinada pelo comprimento da palavra do computador e pode, portanto, variar largamente de máquina para máquina. Por outro lado, é necessário pensar que nem todas as máquinas operam necessariamente sobre a representação binária dos números.

Com efeito, os circuitos electrónicos utilizados nos computadores têm apenas dois estados estáveis (correspondentes, por exemplo, a dois valores da tensão eléctrica), que se notam, por convenção, 0 e 1, de modo que a aritmética mais corrente nos computadores é binária; no entanto, é possível utilizar uma representação interna *decimal codificada em binário* (binary coded decimal BCD), representando em binário cada algarismo decimal de um número por meio de quatro bits.

Exemplo 1.2 Codificação BCD

$3141_{(10)} = 0011_{(2)}0001_{(2)}0100_{(2)}0001_{(2)}$

Exemplo 1.3 Máquina virtual

Em uma máquina (que não existe, mas que vamos utilizar intensivamente daqui em diante) com uma representação decimal com três dígitos na mantissa e um no expoente, cuja representação interna seria da forma: $m \times 10^p$ com $0.1 \leq m \leq 1$. Teríamos, por exemplo, $10 = 0.100 \times 10^2$ e $2^{\frac{1}{2}} = 0.141 \times 10^1$. Deste modo, existem nesta máquina tantos números entre 0.100×10^{-3} e 0.999×10^{-3} como os que há entre 0.100×10^3 e 0.999×10^3, quando é perfeitamente evidente que este último intervalo é muito maior.

Exercício 1.2 Reportando-se ao exemplo anterior, procure responder:

1. Quantos são esses números?
2. Qual a diferença entre dois números consecutivos, em cada um dos casos?
3. Qual é o menor intervalo entre números representáveis sucessivos? e o maior?

Uma medida da máxima precisão atingida pelo cálculo em vírgula flutuante é dado pela *unidade da última casa (u)*, definida como o menor número positivo tal que $u + 1 > 1$.

Exemplo 1.4 Unidade da última casa

O seguinte programa BASIC determina o valor de u em aritmética de precisão simples, diminuindo o valor de $u + 1$ até não ser reconhecido pelo computador como maior que a unidade:

```
U=0.00001
DO
U=0.9999*U
LOOP WHILE (U+1) > 1
PRINT "U = ";U
```

Corrido em um PC-AT em BASICA em precisão simples dá $u = 5.96E-8$, isto é, 24 bits na mantissa, mas corrido em TURBOBASIC dá $u = 2.71E-20$, isto é, 64 bits na mantissa. Com efeito, o valor da unidade da última casa relaciona-se directamente com o número de bits usados na representação dos números em vírgula flutuante. Os PC usam 32 bits (4 bytes) para os números em precisão simples, dos quais 24 são atribuídos à mantissa, incluindo o sinal, e 8 ao expoente, incluindo também o sinal. Assim, o valor de u será $u_s = 2^{-24} = 5.96 \times 10^{-8}$ Em dupla precisão, o PC usa 64 bits (8 bytes) para representar os números em vírgula flutuante, dos quais 56 para a mantissa e 8 para o expoente. O valor de u em dupla precisão será então $u_d = 2^{-56} = 1.39 \times 10^{-17}$.

Exercício 1.3

1. Assegure-se que compreendeu claramente o funcionamento do algoritmo; em caso de necessidade, execute-o em modo "trace".
2. Se é verdade o que acima se disse sobre o PC, como é possível que o TURBOBASIC tenha o comportamento descrito?
3. Como procederia para determinar a unidade da última casa em uma máquina de calcular não-programável?
4. Seria capaz de escrever um programa BASIC para determinar a unidade da última casa em dupla precisão?
5. E em PASCAL ou C, para simples e dupla precisão?
6. E um programa para determinar o maior número representável?

Assim, o sistema de vírgula flutuante é nitidamente diferente dos sistemas habituais dos números da matemática, e um utilizador deve ficar prevenido de que essa diferença pode, por vezes, produzir efeitos surpreendentes.

Uma maneira possível de olhar os números da computação consiste em pensar que cada resultado de um cálculo surge, na representação computacional, um pouco preciso ou esborratado em torno do valor que matematicamente *deveria* ter. Este ponto de vista enquadra-se bem com a ideia que correntemente temos do uso prático dos números mesmo fora do contexto computacional (quando dizemos `uma tonelada de areia` nunca pretendemos seriamente que seja a quantidade exacta até ao último grão e, mesmo que o fosse, a precisão não seria absoluta, sendo apenas da ordem de 10^{-9}). Um outro ponto de vista possível consiste em pensar que os únicos

números existentes são os representáveis na máquina e que não existem mais nenhuns, o que corresponde a considerar que os números reais dos matemáticos são, para o programador e o analista numérico, inteiramente fictícios. Este ponto de vista pode parecer um tanto drástico, mas tem-se revelado extremamente útil ao analista numérico, nomeadamente quando tem que identificar os mecanismos que conduziram a um resultado errado ou simplesmente surpreendente.

Tipicamente, a finitude do intervalo coberto pelo expoente p pode dar origem aos acidentes conhecidos por *underflow* e *overflow* e que correspondem ao facto de o resultado de uma operação exceder o máximo ou o mínimo dos números representáveis na máquina. É relativamente comum - mas perigoso - pensar que não haverá mal em substituir um *underflow* por zero, salvo se, a seguir, entrar como divisor em uma divisão. Pelo contrário, em termos de senso comum, parece perigoso substituir um *overflow* pelo maior número representável na máquina (que não é, em geral, espantosamente grande) e, no entanto, muitos compiladores para uso científico fazem-no, sem sequer avisar o utilizador, para lhe poupar numerosas mensagens de erro ou paragens intempestivas do programa. Esta possibilidade não deve nunca ser perdida de vista, e um utilizador cuidadoso investiga, antes de mais nada, o comportamento de um compilador ou interpretador perante os *overflow* e *underflow* a fim de poder tomar as precauções adequadas para evitar surpresas desagradáveis.

Exercício 1.4 O **`PASCAL`** define a primitiva **`MAXINT`**, que permite conhecer qual o maior valor inteiro positivo que a implementação concreta permite. Primitivas que permitem conhecer o *comprimento* de um objecto, permitem obter informação semelhante, desde que se saiba a *unidade de medida*!. Os mesmos mecanismos não são aplicáveis em relação à representação em vírgula flutuante. Procure identificar, para as linguagens de programação que conhece, quais os mecanismos disponíveis para conhecer os limites das representações de números.

Exemplo 1.5 Overflow

Na máquina hipotética do exemplo anterior, o maior número representável seria $0.999 \times 10^9 \approx 10^9$ enquanto o menor (em valor absoluto, com exclusão do zero) seria $0.100 \times 10^{-9} = 10^{10}$.
Quando é que a exponencial daria *overflow* na nossa máquina hipotética?
Quando fosse

$$\exp(x) > 0.999 \times 10^9$$

isto é, tomando logaritmos,

$$x > \ln 0.999 + 9 \times \ln 10 = -0.001 + 9 \times 2.303 = 20.7$$

os valores são

$$-0,00100050033358353350014298225406 83$$
$$+9\times 2,30258509299404568401799145468 44$$
$$=20,72226533661282762266178010990 5$$

Exercício 1.5 Qual é o valor correspondente para:

1. a sua máquina de calcular?
2. para os diferentes **BASIC** do PC em simples e dupla precisão?
3. neste caso, dependerá da linguagem?

O raciocínio parece perfeitamente simples e claro; tem a certeza de que é geralmente correcto?

No entretanto, temos o direito de nos interrogar porque é que um intervalo representável que, nas piores condições, nunca será inferior a

$$(-10^{38},-10^{-38}),(10^{-38},10^{38})$$

pode com tanta frequência dar origem a problemas de *overflow* ou de *underflow*, quando rarissimamente os resultados dos problemas excedem os limites desses intervalos. A resposta óbvia é a de que tais acidentes surgem habitualmente nos resultados de cálculos intermédios. Assim, a precisão do resultado de um cálculo numérico fica altamente dependente do caminho para a ele chegar, isto é, do algoritmo adoptado para o implementar, o que leva o analista numérico a ter que tomar precauções que são inteiramente desconhecidas para o matemático convencional (nas matemáticas convencionais, a única situação comparável que nos ocorre é a da soma de séries simplesmente convergentes, que depende da ordem das parcelas).

Exemplo 1.6 overflow e underflow

No cálculo de

$$1-\frac{1}{e^x+1}$$

para $x \geq 20.7$ (caso em que o valor da expressão seria 1 quando representado com a precisão da nossa máquina) teríamos *overflow*. A substituição do valor impossível pelo maior número representável na máquina daria o resultado final esperado, 1. Por outro lado, se utilizássemos a expressão matematicamente equivalente

$$\frac{1}{1+e^{-x}}$$

teríamos *underflow* e a sua substituição por zero daria também como resultado final o 1 esperado.

Exercício 1.6

1. Quantos números diferentes podem ser representados em vírgula flutuante na nossa máquina hipotética?
2. Explique o sentido da expressão "*nos computadores, o número 0 está relativamente isolado dos seus vizinhos*". Os seus vizinhos está-lo-ão igualmente entre si? Que influência tem o comprimento da mantissa no fenómeno? e o da característica? Em termos comparativos, que se passa com o maior número representável?
3. Estabeleça as semelhanças e diferenças entre o uso da representação de números em vírgula flutuante e a representação logarítmica.

1.2.5 Representação IEEE

A representação informática de números em vírgula flutuante foi normalizada pelo organismos IEEE [12] e ANSI, definindo regras de representação e de cálculo.

Assim, um número em vírgula flutuante $mantissa \times 2^{expoente}$ deverá ser escrito na forma:

$$\pm 1.\mathtt{f} \times 2^{\mathtt{e}}$$

em que `f` é a parte fraccionária da mantissa, e sendo `e` obtido por

$$e = expoente + vies$$

O padrão define dois tipos:

Número em precisão simples Usa 32 bits repartidos da seguinte forma:

- bit `0`, um bit de sinal;
- bit `1` a `8`: oito bits de expoente (em excesso de 127);
- bit `9` a `31`: vinte e três bits para a parte fraccionária da mantissa, normalizada de maneira à parte inteira ser unitária;

Número em precisão dupla Usa 64 bits repartidos da seguinte forma:

- bit `0`, um bit de sinal;
- bit `1` a `11`: onze bits de expoente (em excesso de 1023);
- bit `12` a `63`: 52 bits para a parte fraccionária da mantissa, normalizada de maneira à parte inteira ser unitária;

[12] Institute of Electrical and Electronics Engineers

Na apresentação que se segue discutiremos apenas os números em precisão simples. A discussão dos números em dupla precisão apenas necessita de ser ajustada ao seu maior tamanho.

O padrão prevê os seguintes valores especiais:

- $+0$, -0 fazendo $e = 0$, com o sinal adequado;
- $-\infty$, $+\infty$ fazendo $e = 255$, com o sinal adequado;
- **`NaN`** fazendo $e = 255$ e $f \neq 0$; ***Not a Number*** é o resultado, por exemplo, da avaliação de $0/0$.

Como o expoente `e` está limitado ao intervalo `]0,255[`,
a potência do número $p \in [-126, +127]$ pelo que o menor número positivo em precisão simples é

$$\begin{aligned} 1.00000000000000000000000 \times 10_2^{-01111110} \\ = 2_{10}^{-126} \approx 1.2 \times 10^{-38} \end{aligned}$$

e o maior número positivo em precisão simples é

$$\begin{aligned} 1.11111111111111111111111 \times 10_2^{+11111110} \\ = (2 - 2^{-23}) \times 2_{10}^{+127} \approx 3.4 \times 10^{+38} \end{aligned}$$

Precisão

Qual a precisão numérica desta representação? O bit menos significativo de `f` tem peso 2^{-23}, pelo que uma mudança nesse bit causa uma variação em `f` igual a 2^{-23}. Consideremos um intervalo $[x_1, x_2]$, definido por dois números representáveis exactamente em precisão simples IEEE e consecutivos, por exemplo:

$$\begin{aligned} x_1 &= 1.00000000000000000000000 \times 10_2^p \\ x_2 &= 1.00000000000000000000001 \times 10_2^p \end{aligned}$$

Como x_1 e x_2 são *números máquina*, representáveis exactamente na notação escolhida, e são consecutivos, não pode ser representado nenhum outro número entre eles. Qualquer quantidade entre x_1 e x_2, ou é representada por x_1 ou por x_2 [13]. Qualquer que seja a escolha, o erro cometido não excederá 2^{-23} no valor de f e 2^{-23+p} no valor do próprio número.

Se x for o número máquina

$$x = m \times 2^p$$

então o número máquina imediatamente superior é

$$x = (m + 2^{-23}) \times 2^p$$

[13] consoante a técnica de arredondamento usada.

e o número máquina imediatamente inferior é

$$x = (m - 2^{-23}) \times 2^p$$

A diferença (espaçamento) entre os números é

$$x_+ - x = x - x_- = 2^{-23} \times 2^p = 2^{-23+p}$$

sendo o espaçamento relativo entre x_+ e x

$$(x_+ - x)/x \approx 2^{-23} \approx 1.2 \times 10^{-7}$$

O espaçamento relativo é uma constante, aproximadamente igual a 2^{-23}, pelo que pode ser usado como uma medida da precisão de representação.

Arredondamento

O *arredondamento* ocorre sempre que a quantidade a representar não seja um *número máquina*, não sendo representável exactamente, o que obriga a escolher um dos dois *números máquina* enquadrantes. O modo de arredondamento padrão é *arredondamento para o mais próximo*, com *arredondamento para par* (para o número que tem um `0` no seu algarismo menos significativo) em caso de indecisão. Esta técnica de arredondamento garante um erro máximo de meia unidade do algarismo menos significativo.

Valores especiais

Os valores *infinito*, *NaN*, e *zero* tem representação e tratamento especiais:

Infinito é tratado como um número muito grande, sempre que tal fizer sentido; por exemplo, sendo x um número positivo em virgula flutuante:

$$\begin{aligned} +\infty + x &\rightarrow +\infty \\ +\infty \times x &\rightarrow +\infty \\ x/+\infty &\rightarrow +0 \\ +\infty/x &\rightarrow +\infty \end{aligned}$$

o mesmo se aplica ao caso de x negativo e de $-\infty$;

NaN Este valor é usado no caso de operações de resultado indeterminado:

$$\begin{aligned} 0/0 &\rightarrow \texttt{NaN} \\ (+\infty) - (+\infty) &\rightarrow \texttt{NaN} \\ x + \texttt{NaN} &\rightarrow \texttt{NaN} \end{aligned}$$

zero O padrão prevê duas formas de zero:

- $+0$ que é o resultado da maioria das operações de que resulta zero;
- -0 que é usado como indicador de um número negativo muito pequeno, inferior à precisão máquina.

Excepções

O padrão define vários tipos de excepções detectáveis, de que importa destacar o *overflow*, o *underflow* e *inexact*, que ocorre quando um resultado deixou de ser exacto em consequência de arredondamentos.

Exercício 1.7 Na definição do padrão IEEE diz-se que a normalização do número deve ser tal que satisfaça

$$\pm 1.\mathtt{f} \times 2^{\mathtt{e}}$$

enquanto que anteriormente tinhamos sugerido que deveria ser

$$\pm 0.\mathtt{f} \times b^{\mathtt{e}}$$

sendo neste caso o algarismo mais significativo de `f` diferente de zero.
Porquê ?
(sugestão: pense na base!)

1.3 Erros

A representação finita (aproximada) de números em um dispositivo computacional (calculador humano, máquina de calcular mecânica ou electrónica, computador digital) conduz inevitável e sistematicamente ao aparecimento de *erros de arredondamento*[14], enquanto a representação finita de processos conduz ao aparecimento de *erros de truncatura*[15]; esta é a nomenclatura convencional, no entanto, dado que toda a representação de um número é um processo característico do cálculo numérico, e, portanto, enquanto processo finito é um processo de truncatura; assim, arredondamento e truncatura são, em termos práticos, essencialmente a mesma coisa.

O *erro absoluto* pode ser definido como a diferença entre o valor exacto de uma quantidade e o seu valor aproximado:

$$e(x) = x_{exacto} - x_{aproximado}$$

O *erro relativo* é a relação

$$r(x) = \frac{e(x)}{x_{exacto}}$$

Observe-se que, das duas equações, se pode deduzir

$$x_{exacto} = \frac{x_{aproximado}}{1 - r(x)}$$

expressão que, caso se conheça um majorante para o erro relativo e que este seja suficientemente pequeno, nos permite obter um intervalo enquadrante do valor exacto.[16]

[14] erro devido à aproximação computacional
[15] erro devido à aproximação matemática
[16] Note-se que a dificuldade em satisfazer essas duas exigências torna a expressão praticamente inútil!

1.4 Arredondamento e Truncatura

Quando se procede à *multiplicação* de dois números com três casas decimais, o produto tem seis casas decimais; se se quiser continuar a ter apenas as três casas decimais dos dados, é de boa prática proceder a um *arredondamento*, isto é, juntar 5 à primeira casa decimal a abandonar, antes de a abandonar; o resultado é que, se essa primeira casa a abandonar for 5 ou superior, teremos um aumento de uma unidade na última casa conservada (*arredondamento para cima*); se, porém, for inferior a 5, o valor da última casa conservada manter-se-á inalterado (*arredondamento para baixo*).

Deste modo, qualquer que seja o caso, o valor arredondado será mais próximo do valor exacto que o que obteríamos mediante uma simples truncatura, isto é, eliminando pura e simplesmente as casas decimais em excesso. Seja como for, cometeremos, em geral, pequenos erros: *erro de arredondamento* e *erro de truncatura*. A importância da pequena diferença entre os dois erros resulta da possibilidade de vir a acumular-se em operações sucessivas e de vir a ser amplificada pelo efeito de operações posteriores, como veremos.

Exemplo 1.7 Multiplicação

Multiplicando dois valores

$$\begin{array}{r} 0.236\times 10^{1} \\ \times\ \ 0.127\times 10^{2} \\ \hline 1652 \\ 472 \\ 236 \\ \hline 0.029972\times 10^{3} \end{array}$$

Normalizando e calculando o valor para arredondar

$$\begin{array}{r} 0.29972\times 10^{2} \\ +\ 0.0005\ \times 10^{2} \\ \hline 0.30022\times 10^{2} \end{array}$$

Podemos verificar na tabela seguinte os valores obtidos por arredondamento e por truncatura e quais os erros cometidos.

×	arredondamento	truncatura
Valor	0.300×10^{2}	0.299×10^{2}
Erro absoluto	-0.280×10^{-1}	0.720×10^{-1}
Erro relativo	0.934×10^{-3}	0.240×10^{-2}

A mesma situação ocorre, naturalmente, na *divisão*. Note-se, de passagem, que as mesmas duas técnicas podem ser utilizadas na conversão de um dado (em geral sob forma decimal) na base própria da máquina.

Na própria *adição* podem ocorrer erros de arredondamento e de truncatura, mesmo quando as características dos números a adicionar são as mesmas, desde que possa haver transporte da casa mais significativa; no caso de os dois números serem de ordens de grandeza muito

diferentes, a adição pode nem sequer ter qualquer efeito, no sentido de que o resultado será, pura e simplesmente, igual ao de maior valor absoluto.

Exemplo 1.8 Adição

Somando dois valores

$$\begin{array}{r} 0.742\times10^{2} \\ +\ 0.927\times10^{2} \\ \hline 1.669\times10^{2} \end{array}$$

Normalizando e calculando o valor para arredondar

$$\begin{array}{r} 0.1669\times10^{3} \\ +\ 0.0005\times10^{3} \\ \hline 0.1674\times10^{3} \end{array}$$

Podemos verificar na tabela seguinte os valores obtidos por arredondamento e por truncatura e quais os erros cometidos.

+	arredondamento	truncatura
Valor	0.167×10^{3}	0.166×10^{3}
Erro absoluto	0.100×10^{0}	0.900×10^{0}
Erro relativo	0.599×10^{-5}	0.539×10^{-4}

No caso de dois números de grandeza diferente, as *mantissas* não podem adicionar-se directamente, pelo que se torna necessária uma operação prévia de *alinhamento* que consiste em reduzir os dois números ao mesmo expoente; após a realização da adição das *mantissas*, o resultado é arredondado, o que implica perda de precisão que pode ir ao ponto de a soma ser igual à maior das parcelas, o que, em si, não será extremamente grave, salvo no caso de adições encadeadas, como sucede na soma de séries.

Exemplo 1.9 Soma com perda de parcela

Para efectuar a soma:

$$0.742 \times 10^{2} + 0.927 \times 10^{-2}$$

é necessário efectuar primeiro o alinhamento de expoentes

$$\begin{array}{r} 0.742 \qquad \times10^{2} \\ +\ 0.0000927\times10^{2} \\ \hline 0.7420927\times10^{2} \end{array}$$

O valor renormalizado é 0.742×10^{2}

Porém, onde os erros deste tipo causam mais problemas é na *subtracção* de números da mesma ordem de grandeza: em consequência da anulação dos primeiros dígitos da mantissa, esta é deslocada para a esquerda até ao primeiro dígito não nulo, o que pode trazer bem para a esquerda os erros de truncatura e/ou de arredondamento. A rigor, neste caso, não há que falar em erro de

arredondamento nem de truncatura, mas sim em erro intrínseco da própria técnica operatória quando realizada sobre dados arredondados ou truncados.

Exemplo 1.10 Subtracção

$$\begin{array}{r} 0.314\times 10^{1} \\ -\ 0.313\times 10^{1} \\ \hline 0.001\times 10^{1} \end{array}$$

O valor renormalizado é 0.100×10^{-1}

Erros absoluto e relativo da operação: 0

Poderia pensar-se que, tal como no caso da adição, este efeito não tenha importância excepcional; porém, salvo se os valores dos termos forem exactos (isto é, dados iniciais ou resultados parciais sem erro) os dois últimos zeros da mantissa são artificiais e nem sequer temos a certeza do 1 que os antecede; com efeito,

- se o aditivo exacto fosse 0.3135×10^{1} e o subtractivo exacto fosse 0.3134×10^{1} , o resultado correcto seria 0.1×10^{-2} , um décimo do valor calculado 0.1×10^{-1};
- se o aditivo exacto fosse 0.3144×10^{1} e o subtractivo exacto fosse 0.3125×10^{1}, o resultado correcto seria 0.19×10^{-1} , dezanove vezes o valor calculado.

Exercício 1.8

1. calcule os erros relativos de um e outro resultados e compare-os com os dos termos da subtracção;
2. repita o raciocínio para o caso de dois termos iguais, tais que o resultado calculado da subtracção é nulo (este é o caso extremo que dá ao fenómeno o nome de cancelamento); imagine as consequências se este resultado entrasse em seguida como factor de uma multiplicação por um número grande o como divisor em uma divisão.

Este fenómeno de *cancelamento* constitui a principal fonte do erro numérico em cálculo científico, não criando erros por si próprio, mas actuando como um amplificador de erros preexistentes. Na maior parte dos cálculos instáveis, criam-se erros numéricos significativos por acumulação de arredondamentos e esses erros são depois amplificados pelo cancelamento. Esta situação é de tal modo frequente e grave que Lieberstein [Lie68] se refere a ela como

> a dificuldade patológica da análise numérica: enormes perdas de precisão resultantes da subtracção de representações finitas de números grandes quando a sua diferença é pequena.

Exemplo 1.11 I.N.Génuo

Retomemos o caso do senhor I.N. Génuo e calculêmo-lo usando uma folha de cálculo, com precisões sucessivamente crescentes. Os resultados obtidos foram os seguintes:

algarismos significativos	capital acumulado
3	-4,375E+21
4	2,8188E+20
5	-2,83618E+19
6	2,660808E+18
8	2,39020731E+16
10	6,3525795660E+14
12	-7,016551748043E+11
13	8,4946582952843E+11
14	7,39053273591465E+10
15	7,390532735914650E+10
16	7,3905327359146500E+10

Não se entusiasme com a aparente estabilização (numa considerável fortuna!). Acontece que a folha de cálculo utilizada (MS Excel) tem uma precisão máxima de 14 algarismos significativos!
Se pretender explorar o exemplo, pode recorrer ao ficheiro Ingenuo.xls .

Exercício 1.9 Calcule o problema do senhor I.N. Génuo com precisões ainda maiores. Investigue se o valor tenderá realmente a estabilizar.
(sugestão: utilize um manipulador numérico ou algébrico com precisão estipulável, como o DERIVE)

Exemplo 1.12 Transformação de expressões

No cálculo de $(x+1)^{1/2}-x^{1/2}$ não há problemas de maior quando x é pequeno, mas pode ocorrer importante perda de precisão quando x é grande. Neste caso, pode com vantagem proceder-se à transformação

$$\frac{[(x+1)^{1/2}-x^{1/2}].[(x+1)^{1/2}+x^{1/2}]}{(x+1)^{1/2}+x^{1/2}} = \frac{1}{(x+1)^{1/2}+x^{1/2}}$$

Podemos verificar os resultados executando a seguinte sequência de comandos no Maple:

```
> Digits:=5;
```

$$Digits := 5$$

```
> f(x) := (x+1)^(1/2)-x^(1/2);
```

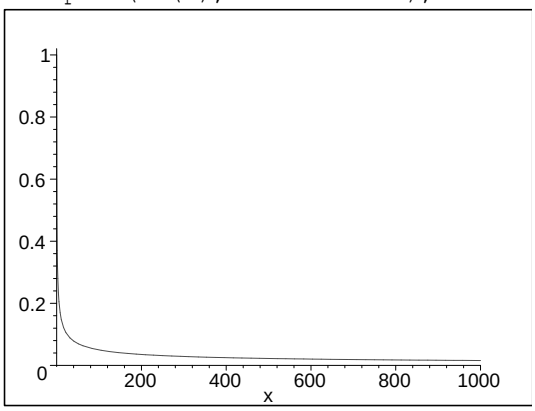
$$f(x) := \sqrt{x+1}-\sqrt{x}$$

```
> f1(x):= 1/((x+1)^(1/2)+x^(1/2));
```

$$\mathrm{f1}(x) := \frac{1}{\sqrt{x+1}+\sqrt{x}}$$

Gráficos das duas funções

```
> plot(f(x) ,x= 0..  + 10^3);
> plot(f1(x),x= 0..  10^3);
```



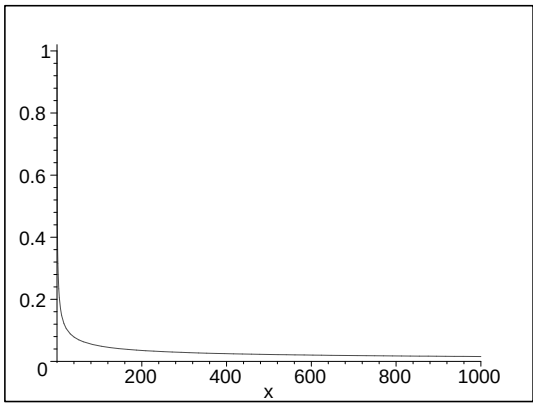
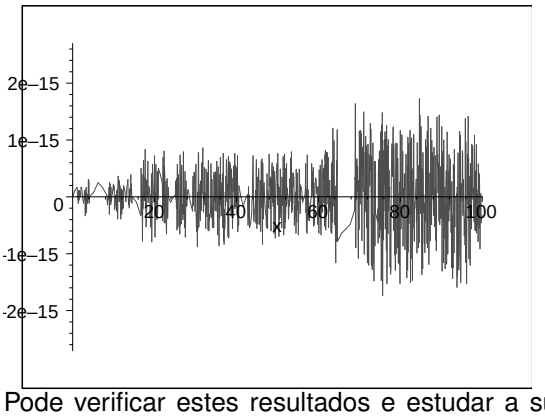


Gráfico da diferença das duas funções

```
> plot(f(x) -f1(x),x= 0..  + 1.0E2);
```



Pode verificar estes resultados e estudar a sua variação com o número de dígitos de precisão e com o valor de x usando o ficheiro Perda de Precisao.mws .

Uma vez cometido um erro de arredondamento no resultado de uma operação, esse erro vai contaminar os dados das operações subsequentes segundo um mecanismo de *propagação do erro*, mecanismo que depende do encadeamento particular de operações em cada caso. Existem, naturalmente, numerosos artifícios para diminuir os efeitos perversos do erro de arredondamento. A dificuldade principal é, porém, a de prever quando e onde é que o arredondamento vai criar problemas.

Um bom exemplo do tipo de artifício comparativamente elementar é o relativo à adição (em vírgula flutuante) de um número moderadamente elevado de números relativos: se nos limitarmos a adicioná-los sucessivamente, arriscamo-nos a acumular muitos arredondamentos. Uma maneira de o evitar consiste em separar os somandos em duas classes, a dos positivos e a dos

negativos, ordenando cada uma do maior para o mais pequeno; começa-se então a adição pelo mais pequeno de uma das classes, juntando-lhe sucessivamente membros da outra classe até que o sinal do resultado mude; a cada mudança de sinal do resultado passa-se à outra classe para tomar o número seguinte (imediatamente superior); deste modo, a soma acumulada é mantida tão pequena quanto possível; quando, finalmente, uma das classes se esgota, os números da restante podem ser adicionados ao resultado parcial sem cuidados especiais. Obviamente, este método exige esforço importante na separação e ordenação dos somandos de modo que, sempre que uma máquina disponha de representação de números em dupla precisão, tende a preferir-se o uso desta.

Para a pessoa habituada a calcular à mão com 4 ou 5 casas decimais, um erro de arredondamento na 8^a casa decimal, ou mesmo mais além, pode parecer coisa trivial e inofensiva ou até mesmo excelente, mas o facto é que os enormes encadeamentos de cálculos característicos da computação digital automática podem acarretar acumulações substanciais desses erros. O facto de ser praticamente impossível prever os efeitos de erros de arredondamento e/ou truncatura ao fim de longas e complexas cadeias de cálculos faz com que os espíritos simplistas considerem o processo de acumulação de erros como aleatório, argumentando que a lei dos grandes números torna aproximadamente iguais os casos de erros positivos e negativos, que assim tenderiam a compensar-se. A aparência altamente científica deste argumento não chega para esconder os factos de:

1. a experiência revelar que a existência de correlações (isto é, de dependências) internas entre os sucessivos resultados parciais de um cálculo científico complexo é muito mais comum que o que correntemente se pensa; tais correlações fazem com grande frequência que os erros ocorram dominantemente em um de dois sentidos, o que leva a uma efectiva acumulação;

2. aquilo que, no fundo, está em questão não é a probabilidade de um erro grande (que pode ser, efectivamente, pequena) mas a esperança matemática do custo que lhe está associado (isto é, a soma dos produtos das probabilidades de cada erro pelos respectivos custos), que pode ser insuportável.

Nestas condições, todo o cuidado é pouco nesta matéria e as belas folhas de números impressos pelo computador só são razoavelmente fiáveis quando forem tomadas todas as precauções possíveis. É este facto que nos leva, na perspectiva da computação automática, a consagrar tanto espaço de um curso de análise numérica ao problema dos erros.

Exemplo 1.13 Expressões factorizadas e expandidas

Consideremos a expressão

$$\begin{aligned}(5x-14)^2(x^2+3) = \\ &= 25x^4-140x^3+271x^2-420x+588 \\ &= 25x^4+271x^2+588-140x^3-420x\end{aligned}$$

e vejamos como o Maple calcula estas diferentes expressões alternativas com 11 algarismos significativos nas vizinhanças da única raiz positiva, $x = 2.8$.

```
> Digits:=11;
```

$$Digits := 11$$

```
> par:= (x,y)->[x,y];
```

$$par := (x, y) \rightarrow [x, y]$$

Construção das expressões:

```
> ff:= x->(5.0*x-14.0)^2*(x^2+3.0);
> fexp:= expand(ff(x) );
> fd1 := x->25.0*x^4+271.0*x^2+588.0-140.0*x^3-420.0*x;
```

$$ff := x \rightarrow (5.0x - 14.0)^2 (x^2 + 3.0)$$

$$fexp := 25.00x^4 + 271.000x^2 - 140.00x^3 - 420.000x + 588.000$$

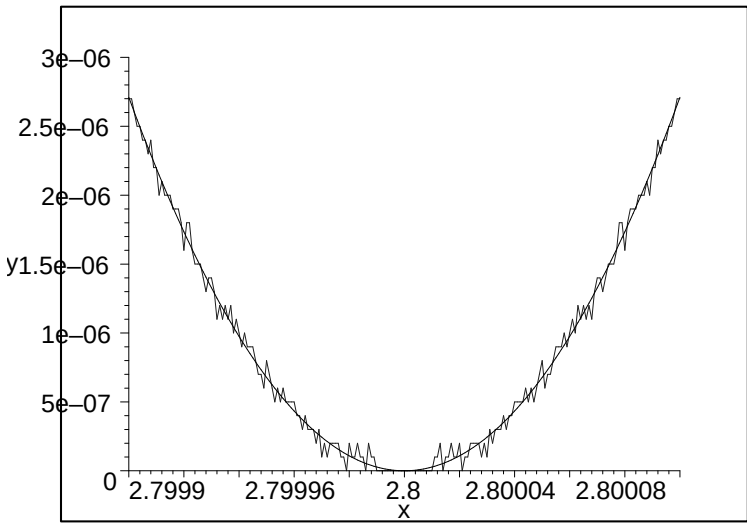
$$fd1 := x \rightarrow 25.0x^4 + 271.0x^2 + 588.0 - 140.0x^3 - 420.0x$$

Construção das listas de amostragem em x

```
> lx:= [seq(0.000001*i,i=2799900..  2800100)];
> lff:= [seq(ff(x) , x= lx)];
> lfd1:= [seq( fd1(x), x= lx)];
> lxff:= zip(par,lx,lff);
> lxfd1:= zip(par,lx,lfd1);
```

Gráfico da forma factorizada fct(x) e da primeira forma expandida fd1(x)

```
> plot([lxff ,lxfd1], x=2.7999 ..
> 2.8001,y=0 ..3e-6, symbolsize= 4,
> style = line, adaptive=false, color=[black,blue]);
```



(veja o ficheiro Ordem-parcelas.mws)

Exercício 1.10

1. Discuta os méritos relativos de dispor de subrotinas de biblioteca para calcular $\arctan(x)$ e $\arcsin(x)$ por desenvolvimento em série de Taylor; proponha um critério de truncatura para esses desenvolvimentos; discuta os méritos relativos do cálculo das restantes funções trigonométricas inversas a partir de uma destas por meio de fórmulas adequadas.

2. Tendo em consideração o carácter discreto do conjunto dos números representáveis em vírgula flutuante, discuta os problemas relativos ao cálculo de $\text{sen}(x)$ por desenvolvimento em série de Taylor.

1.4.1 Diferentes maneiras de calcular uma expressão

Dada uma expressão analítica $f(x)$ de uma função, pode considerar-se que ela especifica um algoritmo de cálculo dos valores da variável dependente para cada valor da variável independente, x; usa-se então o termo *condição* (cf. [DB74]) para descrever a sensibilidade do valor calculado de $f(x)$ a variações do argumento, x. A condição pode, portanto, ser medida pela variação (relativa) máxima de $f(x)$ em termos da correspondente variação (relativa) de x, isto é, por

$$cond[f(x)\,] = max_{x^*} \frac{\left|\frac{f(x)-f(x^*)}{f(x)}\right|}{\left|\frac{x-x^*}{x}\right|}$$

em que x^* é a representação aproximada de x; se $f(x)$ puder ser desenvolvida aproximadamente na forma

$$f(x) - f(x^*) \cong f'(x).(x-x^*)$$

para pequenos valores de $|x-x^*|$ pode dar-se a esta expressão a forma mais simples:

$$cond[f(x)\,] \cong \left|\frac{x.f'(x)}{f(x)}\right|$$

No entanto, esta forma aproximada deve ser usada com grande cuidado porque, como se afirmou atrás, o modo e a ordem por que se realizam as diferentes operações em uma expressão não é de modo nenhum indiferente do ponto de vista do erro cometido e esta expressão aproximada não toma esse facto em consideração. Por outro lado, é muito pouco interessante o facto de, nesta aproximação, a condição se tornar infinita nas vizinhanças de uma raiz da expressão.

Exemplo 1.14 Cálculo da condição

Retomemos o caso do polinómio do 4^o grau do exemplo anterior e calculemos a sua condição, usando para isso o Maple.

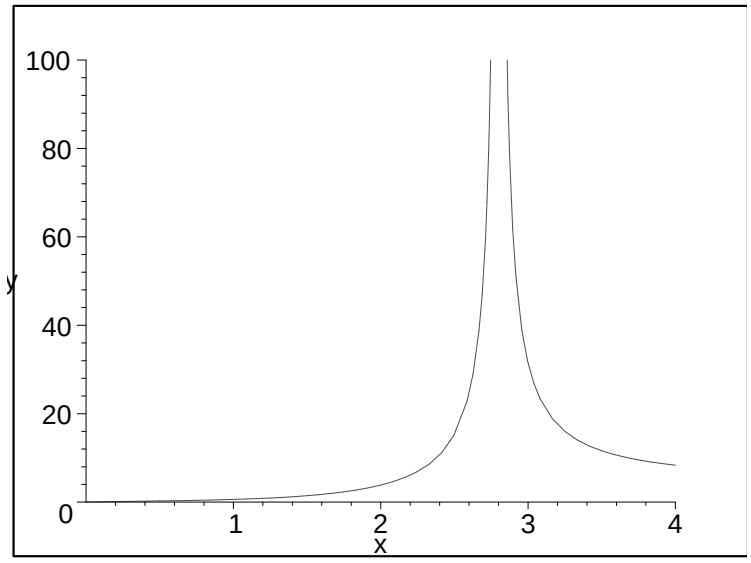
```
> ff(x) :=(5*x-14)^2*(x^2+3);
```

$$\text{ff}(x) := (5x-14)^2\,(x^2+3)$$

```
> condff(x) :=abs((x*diff(ff(x) ,x))/ff(x) );
```

$$\text{condff}(x) := \left| \frac{x\,(10\,(5x-14)\,(x^2+3)+2\,(5x-14)^2\,x)}{(5x-14)^2\,(x^2+3)} \right|$$

```
> plot(condff(x) ,x=0 ..  4, y=0 ..100);
```



No gráfico da condição da forma factorizada do polinómio do exemplo anterior. Observemos o crescimento desmesurado da condição a partir de x = 2.5, precisamente a zona da raiz onde atrás observámos os estranhos fenómenos de perda de precisão. Verificaremos também sem dificuldade que a condição das três expressões diferentes da mesma função é a mesma, como seria de esperar. (veja o ficheiro Condição.mws)

Analisemos agora alguns casos exemplares desta situação: mesmo nos casos mais simples podem ocorrer situações de perda significativa de precisão, nomeadamente em casos relacionados com o já mencionado comportamento peculiar da subtracção. Embora pareça haver uma grande variedade de artifícios para tratar este tipo de problema, o leitor não terá, mediante um pouco de reflexão, grande dificuldade em identificar a maioria como variações do tipo geral de artifício necessário para calcular derivadas a partir da definição. Para evitar os percalços ligados a este tipo de situações, torna-se necessário apenas o uso da imaginação e o cuidado de procurar prever o que pode vir a suceder, antes de passar à fase de programação; a regra fundamental é a de, sempre que possível, evitar subtracções, mesmo que apareçam sob a forma de somas de termos de sinais diferentes. Os exemplos juntos são apenas de alguns dos métodos que o leitor deve já ter encontrado em álgebra e em análise matemática.

Exemplo 1.15 Expressões alternativas

No cálculo de

$$\frac{\text{sen}\,(x+\delta x)-\text{sen}\,(x)}{\delta x}$$

para pequenos valores de x, há toda a vantagem em utilizar a expressão alternativa

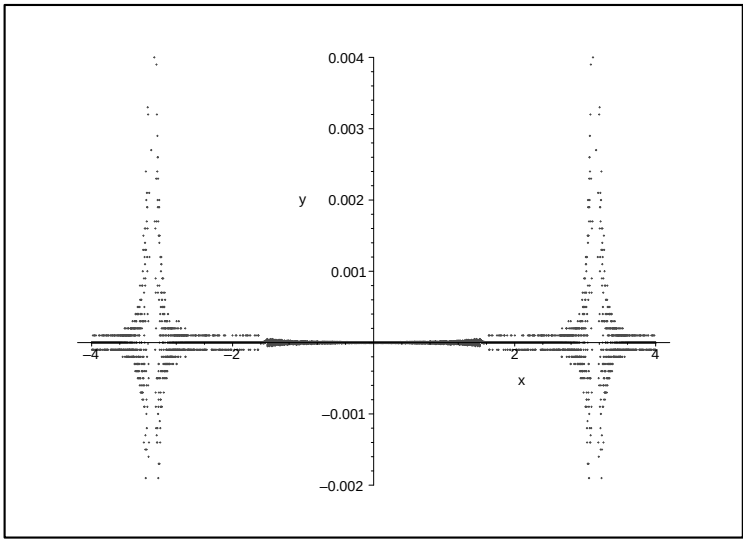
$$\cos\left(x+\frac{\delta x}{2}\right).\frac{\operatorname{sen}\left(\frac{\delta x}{2}\right)}{\frac{\delta x}{2}}$$

Do mesmo modo,

$$1-\cos(\delta x)=\frac{\operatorname{sen}^2(\delta x)}{1+\cos(\delta x)}=2.\operatorname{sen}^2\left(\frac{\delta x}{2}\right)$$

No entanto, é necessário o máximo cuidado: uma boa aproximação para pequenos valores de x pode ser desastrosa para valores altos, como mostra o gráfico, construído em Maple:

```
> Digits:= 5;
> s:= seq(0.001*i,i=-4000..  4000);
> p:= [seq([dx, 1-cos(dx)-((sin(dx)^2)/(1+cos(dx)))], dx=[s])];
> plot(p,x=-4.2 ..  4.2,y=-0.002 ..  0.004,
> symbolsize= 5, style = point,
> adaptive=false);
```



Porém, nem sempre simples rearranjos das expressões bastam para produzir novas expressões satisfatórias para o uso em cálculo numérico: por vezes é necessário recorrer a conhecimentos mais avançados das matemáticas, nomeadamente ao uso de desenvolvimento de funções em séries (de potências ou de outras funções simples). Neste caso, um critério simplista seria o de calcular e somar todos os termos (necessariamente decrescentes em valor absoluto a partir de certa ordem) que o cálculo numérico não apresente como nulos dentro da precisão utilizada; notar-se-á, porém, que

- no caso das séries de termos de sinais alternados, esse critério parece ter, para além da simplicidade, a vantagem de fixar o valor absoluto máximo do erro no limite da precisão da máquina; no caso das séries de termos de sinal constante, tal não se verifica, visto que

o erro - correspondente, ele próprio, a uma série de termos de sinal constante - será em geral muito superior ao valor absoluto do primeiro valor abandonado;

- mesmo no caso mais favorável das séries de termos de sinais alternados, esse critério não só não dá garantias efectivas quanto ao erro como corresponde, em geral, a um esforço de cálculo inutilmente longo, na medida em que já se terá, de há muito, ultrapassado o limite em que as novas contribuições fazem crescer a soma;

Assim, um critério mais eficiente será o de abandonar o cálculo quando, persistentemente, a soma deixar de crescer, mesmo que as parcelas não sejam ainda nulas, o que corresponde ao limite efectivo do cálculo dentro da precisão da máquina, sob a condição de o ordenamento das parcelas ter sido adequado.

Exemplo 1.16 Desenvolvimento em série de Taylor

Seja o desenvolvimento em série de Taylor da exponencial negativa e^{-x} até ao 5^o grau, e o correspondente erro:

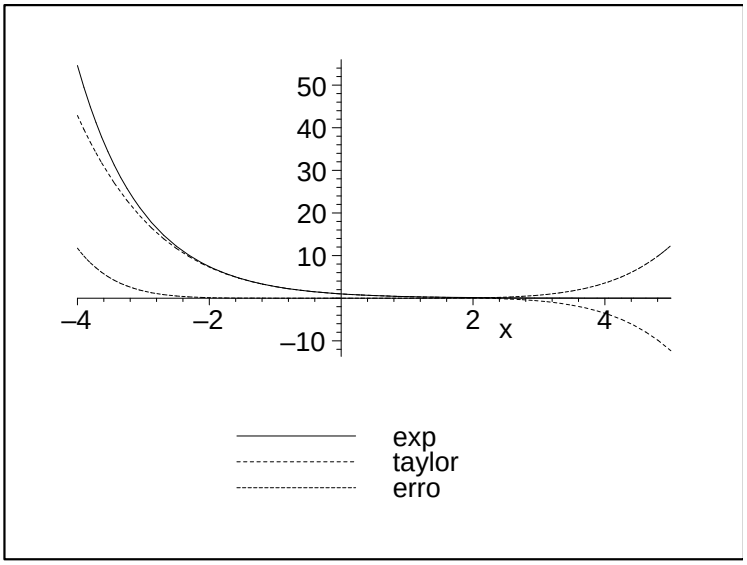
```
> taylor( exp(-x), x=0, 6 );
```

$$1 - x + \frac{1}{2}x^2 - \frac{1}{6}x^3 + \frac{1}{24}x^4 - \frac{1}{120}x^5 + \mathrm{O}(x^6)$$

```
> te(x):= convert(%,polynom);
```

$$\mathrm{te}(x) := 1 - x + \frac{1}{2}x^2 - \frac{1}{6}x^3 + \frac{1}{24}x^4 - \frac{1}{120}x^5$$

```
> plot( [exp(-x), te(x), exp(-x)-te(x)], x=
> -4 ..  5, linestyle= [1,2,3], legend=["exp", "taylor",
> "erro"]);
```



Exercício 1.11 Determine o número de termos do desenvolvimento em série de Taylor da ex-

ponencial negativa que devem ser usados para calcular e^{-1} na nossa máquina hipotética.
O termo geral do desenvolvimento é da forma

$$\frac{(-1)^k.x^k}{k!}$$

Exemplo 1.17 Exponencial e Série de Taylor

O cálculo de $e^{a.x}-1$ para valores relativamente grandes de $|a.x|$ pode, sem grandes problemas, ser abordado pelo cálculo da exponencial pelo método anterior. Porém, para o caso de $|a.x|$ ser pequeno, em que a exponencial se torna próxima da unidade, o erro originado pela subtracção pode tornar-se intolerável; neste caso, é preferível recorrer directamente ao desenvolvimento em série de Taylor da expressão toda

$$e^{a.x}-1=a.x+\frac{(a.x)^2}{2!}+\frac{(a.x)^3}{3!}+\ldots$$

Exercício 1.12 Proponha métodos de cálculo (x qualquer, h pequeno) para as funções:

$$\begin{aligned}
f(x) &= \frac{1}{x+1}-\frac{1}{x}\\
f(x) &= \cos(x+h)-\cos(x)\\
f(x) &= (x+1)^{1/3}-x^{1/3}\\
f(x) &= \frac{1-\cos(h)}{h^2}\\
f(x) &= \ln\frac{1-x}{1+x}
\end{aligned}$$

Exercício 1.13 Discuta os méritos do cálculo de um polinómio

$$p_n(x)=a_n.x^n+a_{n-1}.x^{n-1}+\ldots+a_1.x+a_0$$

na forma:

$$p_n(x)=(\ldots(a_n.x+a_{n-1}).x+\ldots+a_1).x+a_0$$

1.4.2 Os erros nos dados

Uma outra fonte de erro, em que raramente se pensa, mas que tem consequências muitas vezes surpreendentes é a que resulta de os coeficientes das expressões com que se trabalha em Engenharia provirem, directa ou indirectamente, de medidas cuja precisão é finita e, frequentemente, pequena. Esses erros, mesmo minúsculos, podem afectar decisivamente os resultados de um cálculo efectuado sobre os valores correspondentes.

Exemplo 1.18 Erros nos dados

Seja a equação

$$a.x^2 + b.x + c = 0$$

cujos coeficientes a, b, c são determinados experimentalmente e têm os seguintes valores
$a = 1.50 \pm 0.01 \qquad b = -3.45 \pm 0.01 \qquad c = 1.98 \pm 0.01$
Se tomarmos apenas os valores centrais do intervalo temos a equação

$$1.50x^2 - 3.45x + 1.98 \quad \text{cujas raízes são} \qquad \begin{aligned} x_1 &= 1.10 \\ x_2 &= 1.20 \end{aligned}$$

Se, porém, tivermos em conta as variações possíveis dos coeficientes, podemos, para os respectivos valores extremos formar as equações

$$\begin{aligned}
1.51x^2 - 3.46x + 1.99 \quad &\text{cujas raízes são} \qquad x_1 = -1.15 - 0.07i \\
&\qquad\qquad\qquad\quad x_2 = -1.15 + 0.07i \\
1.51x^2 - 3.46x + 1.97 \quad &\text{cujas raízes são} \qquad x_1 = -1.24 \\
&\qquad\qquad\qquad\quad x_2 = -1.06 \\
1.51x^2 - 3.44x + 1.99 \quad &\text{cujas raízes são} \qquad x_1 = -1.14 - 0.14i \\
&\qquad\qquad\qquad\quad x_2 = -1.14 + 0.14i \\
1.51x^2 - 3.44x + 1.97 \quad &\text{cujas raízes são} \qquad x_1 = -1.14 - 0.08i \\
&\qquad\qquad\qquad\quad x_2 = -1.14 + 0.08i \\
1.49x^2 - 3.46x + 1.99 \quad &\text{cujas raízes são} \qquad x_1 = -1.28 \\
&\qquad\qquad\qquad\quad x_2 = -1.05 \\
1.49x^2 - 3.46x + 1.97 \quad &\text{cujas raízes são} \qquad x_1 = -1.32 \\
&\qquad\qquad\qquad\quad x_2 = -1.00 \\
1.49x^2 - 3.44x + 1.99 \quad &\text{cujas raízes são} \qquad x_1 = -1.15 - 0.05i \\
&\qquad\qquad\qquad\quad x_2 = -1.15 + 0.05i \\
1.49x^2 - 3.44x + 1.97 \quad &\text{cujas raízes são} \qquad x_1 = -1.26 \\
&\qquad\qquad\qquad\quad x_2 = -1.05
\end{aligned}$$

o que significa que não só há fortes variações quantitativas nos valores das raízes como há, também, variações qualitativas, visto que existe uma zona de raízes complexas (que, num

contexto físico, representam impossibilidades) e uma outra de raízes negativas (que, em certos contextos físicos, podem representar um tipo diferente de situação). Pode mesmo verificar-se que, dentro da zona da indeterminação dos coeficientes, existe uma zona em que as raízes são iguais: trata-se do caso $b = \pm 2.(a.c)^{1/2}$, isto é,

$$a.x^2 + 2.(a.c)^{1/2}.x + c = 0 \quad \text{cujas raízes são} \qquad x_1 = x_2 = -\left(\frac{c}{a}\right)^{1/2}$$

$$a.x^2 - 2.(a.c)^{1/2}.x + c = 0 \quad \text{cujas raízes são} \qquad x_1 = x_2 = \left(\frac{c}{a}\right)^{1/2}$$

Nestas condições, o próprio sentido que possa ter a intenção de obter resultados precisos, mediante cálculo numérico, toma contornos muito nebulosos. Com efeito, sucede frequentemente – como no caso do exemplo acima, que os resultados calculados para diferentes valores dos parâmetros dentro dos respectivos intervalos de variação provável sejam qualitativamente diferentes; neste tipo de situação, a interpretação dos resultados torna-se altamente problemática e só um perfeito domínio do contexto concreto do problema pode permitir avançar neste terreno com um mínimo de segurança. Posto nestes termos, o problema é de natureza semântica e não puramente matemática, circunstância que raramente é apreciada em todo o seu valor e generalidade.

Observe-se, neste contexto, que instabilidades na resolução de equações, provenientes de variações dos seus parâmetros (coeficientes, expoentes, etc.), resultam não só de situações de incerteza externa mas também de efeitos internos da máquina, como os provindos da conversão de dados decimais em forma binária, o que, só por si, pode acarretar erros.

Exemplo 1.19 A base e a exactidão da representação

Pense o leitor na conversão do número decimal (suposto exacto) 0.1 em forma binária e na sua possível representação em vírgula flutuante.

Em binário

$\bowtie$ logo: $0.11 \times 10^{-11}_{(2)} \approx 0.1_{10}$

Em octal

$\bowtie$ logo: $0.631 \times 10^{-1}_{(8)} \approx 0.1_{10}$

Em hexadecimal

$\bowtie$ logo: $0.631 \times A^{0}_{(16)} \approx 0.1_{10}$

Exercício 1.14

1. Tendo em consideração o exemplo anterior, calcule com a precisão da nossa máquina imaginária, as raízes das equações:

$$x^2 - 100x - 4 = 0$$
$$x^2 - 60x + 2 = 0$$

2. Tendo em consideração o exemplo e os exercícios anteriores, discuta a resolução da equação quadrática geral $ax^2 + bx + c = 0$ dos pontos de vista da precisão dos resultados e da economia de cálculo.

1.4.3 Cálculo dos erros

Na impossibilidade de evitar por completo os erros do cálculo numérico, como minimizá-los? Todos os exemplos anteriores mostram a que ponto o resultado de um cálculo pode desviar-se do valor exacto devido aos erros de arredondamento. O uso de máquinas de elevada precisão e o respeito de certas regras básicas da programação numérica limitam as consequências de certos problemas mas não as eliminam totalmente.

Na impossibilidade de eliminar por completo os erros, como estimá-los? Em primeiro lugar note-se que, se fosse possível estimar com razoável precisão o erro de um cálculo, seria, por definição, possível corrigi-lo. Assim sendo, apenas duas vias estão abertas:

- uma é de carácter determinístico, dá um resultado certo: trata-se da chamada *aritmética intervalar*, baseada em uma ideia original de R. E. Moore (Interval Analysis, Prentice-Hall, Englewood Cliffs, 1963) e especialmente desenvolvida nos anos 80 por investigadores da IBM. O seu princípio é relativamente simples: suponhamos que pretendemos adicionar dois números máquina, X e Y (usamos maiúsculas para representar os números máquina e minúsculas para os números "reais"). O resultado dessa soma é, ou número máquina igual ou imediatamente inferior ao valor z da soma real $[Z_1 = \nabla(z)]$, ou o número máquina igual ou imediatamente superior a z $[Z_2 = \triangle(z)]$; assim, em vez de representar o número z por Z_1 ou Z_2 , passaremos a representá-lo pelo intervalo (Z_1, Z_2) ; procedendo sistematicamente deste modo, obteremos, no final, um intervalo exacto no qual o valor real se encontra necessariamente contido. A Universidade de Karlsruhe desenvolveu, a partir de 1986, software eficiente deste tipo; infelizmente, na grande maioria dos casos, os intervalos finais calculados são horrendamente pessimistas.

Exemplo 1.20 Análise de erros

Se somarmos ou subtrairmos duas quantidades x_1, x_2 afectadas de erros δx_1 e δx_2, o resultado aproximado será

$$y = x_1 \pm x_2$$

enquanto o resultado exacto será

$$y + \delta y = x_1 + \delta x_1 \pm (x_2 \pm \delta x_2)$$

de modo que o erro cometido em consequência da aproximação vale

$$\delta y = \delta x_1 \pm \delta x_2$$

Dado, porém, que não podemos conhecer o sinal dos erros, escreveremos apenas

$$|\delta y| \leq |\delta x_1| + |\delta x_2|$$

o que nos permite enunciar a regra de que o módulo do erro absoluto de uma soma (ou diferença) é majorado pela soma dos módulos dos erros absolutos dos operandos.

Para a multiplicação, o resultado aproximado é

$$y = x_1 . x_2$$

e o exacto é

$$y + \delta y = (x_1 + \delta x_1) . (x_2 + \delta x_2)$$

, de modo que o erro absoluto vale

$$\delta y = x_1 . \delta x_2 + x_2 . \delta x_1 + \delta x_1 . \delta x_2$$

e o erro relativo

$$\Delta y \equiv \frac{\delta y}{y} = \frac{x_1 . \delta x_2 + x_2 . \delta x_1 + \delta x_1 . \delta x_2}{x_1 . x_2} = \Delta x_1 + \Delta x_2 + \Delta x_1 . \Delta x_2$$

ou, supondo que os erros relativos dos operandos são significativamente menores que a unidade,

$$\Delta y \cong \Delta x_1 + \Delta x_2$$

Para a divisão, o quociente aproximado é

$$y = \frac{x_1}{x_2}$$

e o quociente exacto

$$y + \delta y = \frac{x_1 + \delta x_1}{x_2 + \delta x_2} = \frac{(x_1 + \delta x_1) . (x_2 - \delta x_2)}{x_2^2 - \delta x_2^2} \cong y + \frac{\delta x_1}{x_2} - y . \Delta x_2$$

e o erro relativo

$$\Delta y \cong \Delta x_1 - \Delta x_2$$

A semelhança das expressões dos erros relativos do produto e do quociente e o facto de não conhecermos o sinal dos erros permite escrever uma expressão única para os dois casos: de onde a regra de que o módulo do erro relativo de um produto (ou quociente) é majorado pela soma dos módulos dos erros relativos dos operandos.

5. Embora estas regras não sejam particularmente difíceis de usar, a sua aplicação a cada passo de um cálculo complexo, como na aritmética intervalar, torná-lo-ia extremamente penoso. Por isso, muitos analistas numéricos estão interessados em simplificar estas regras, mesmo que à custa de alguma precisão. As regras aproximadas servirão para determinar, aproximadamente, o número de algarismos a reter num resultado, com base no número de algarismos exactos dos operandos; tais algarismos

a reter chamam-se *algarismos significativos*. A validade da análise dos algarismos significativos depende, naturalmente, da base de numeração usada, porque desprezar um algarismo em uma base pode corresponder a desprezar dois ou mais em uma base menor; além disso, dois números que, em dada base, têm o mesmo número de algarismos significativos podem, em outra base, ter um número diferente. Assim, devido à possível perda de significado em consequência de a base de numeração eventualmente usada no cálculo não coincidir com aquela em que se apresentam os resultados (como sucede no cálculo electrónico), é de boa prática reter pelo menos um algarismo incerto, que se chama *dígito de guarda*, desprezando-o apenas no resultado final.

6. Da regra anterior relativa ao erro absoluto da soma e da diferença, resulta que esse erro é dominado pelo maior erro dos operandos; na pior das hipóteses, os dois operandos terão erros da mesma ordem, de modo que o erro do resultado será majorado pelo dobro do maior dos erros dos operandos. Por outro lado, poderemos normalmente admitir que o erro na representação de um número não será superior a meia unidade da última casa significativa, de modo que o operando dominante será aquele que tiver menor número de algarismos significativos à direita da vírgula. A regra para a determinação do número de algarismos significativos de uma soma ou diferença será, portanto, a seguinte: o número de algarismos a reter na parte fraccionária do resultado de uma adição (ou subtracção) não será maior que o número de algarismos da parte fraccionária do operando que a tiver menos extensa.

7. De modo semelhante, deduzimos da regra anterior relativa os erros relativos do produto e do quociente que o erro relativo é dominado pelo do operando que tiver maior erro relativo. Para obter uma aproximação em termos de algarismos significativos, utilizamos a representação em vírgula flutuante: $x = m.r^p$, em que a mantissa m não tem zeros imediatamente à direita da vírgula e r é a base de numeração usada; então

$$r^{-1} \leq |m| \leq 1$$

Por outro lado, se x tiver q algarismos significativos, então a sua mantissa, m, terá esse mesmo número de algarismos; logo, a grandeza do erro absoluto de m não será maior que $r^{-q}/2$ de modo que

$$\delta x \leq \tfrac{1}{2}.r^{p-q}$$

Dado que $|m| \geq r^{-1}$, resulta $|x| \geq r^{p-1}$, logo

$$|\Delta x| = \left|\frac{\delta x}{x}\right| \leq \tfrac{1}{2}.r^{1-q}$$

Deste modo, a grandeza do erro relativo de um operando depende apenas do número de algarismos significativos desse operando, donde resulta a regra: o número de algarismos significativos de um produto (ou quociente) é majorado pelo número de algarismos significativos do operando que os tiver em menor número.

8. Esta análise de algarismos significativos é tipicamente útil para o cálculo manual, mas pode utilizar-se em cálculo automático; porém, para este fim, as linguagens con-

vencionais não são convenientes, tornando-se necessário o recurso a linguagens de processamento de listas.

- uma outra via, dita *método de perturbação*, é de carácter probabilístico e começou a ser explorada em meados dos anos 70 na Universidade Pierre et Marie Curie de Paris. O seu princípio é também simples e engenhoso: a melhor maneira de testar o efeito, sobre o resultado final, das perdas de informação por arredondamento consiste em simulá-la, perturbando cada cálculo elementar por vários erros aleatórios da ordem de grandeza dos arredondamentos e estudando estatisticamente os resultado. O Instituto Francês do Petróleo desenvolveu, a partir de 1987, um package deste tipo; infelizmente, o cálculo torna-se excessivamente lento para todas as aplicações práticas excepto as de maior responsabilidade.

Exercício 1.15

1. Calcule $\text{sen}(x)$ para $x = 2rad$ por meio de um desenvolvimento em série de Taylor. Calcule cada termo da série com sete algarismos significativos. Some os termos e compare o resultado com o valor correcto a nove decimais: 0.909297427. Comente.

2. Calcule $\cos(x) - \cos(0.1)$ a sete algarismos significativos para $x = 0$, $x = 0.05$, $x = 0.095$ e $x = 0.995$,

   a) usando uma tábua ou uma máquina de calcular

   b) usando a identidade

   $$\cos(x) - \cos(y) = 2.\text{sen}\left(\frac{y+x}{2}\right).\text{sen}\left(\frac{y-x}{2}\right)$$

   compare os resultados e comente.

1.5 Conclusão

O objectivo do presente capítulo foi, essencialmente, o de chamar a atenção do estudante para o facto de não poder nunca confiar-se cegamente nos resultados fornecidos por um computador e para o facto de os problemas de precisão do cálculo numérico automático serem ainda matéria de investigação científica e tecnológica. Os factos centrais a destacar são:

- o de uma longa sucessão de cálculos poder conduzir a um resultado grosseiramente errado e

- o de as consequências deste primeiro facto poderem ser extremamente graves: os cálculos em vírgula flutuante são utilizados para projectar edifícios, pontes, barragens, automóveis e aviões.

Vimos também, por outro lado, que, antes de pôr em exploração uma nova aplicação, é possível e desejável, embora caro e trabalhoso, testá-la do ponto de vista da precisão.

O grande matemático A. Householder dizia que tinha medo de andar de avião porque sabia que tinha sido projectado usando aritmética de vírgula flutuante; trata-se de um comentário extremamente sério não só a um estado de coisas ainda insatisfatório mas, sobretudo, às práticas descuidadas e irresponsáveis que, infelizmente, são correntes.

[Ham71]
[Lie68]
[Moo63]
[Mul89]
[Wal90]

Bibliografia

[CB81] Conte and De Boor. *Elementary Numerical Analysis, an algorithmic approach*. McGraw-Hill International Editions, Singapore, 1981.

[DB74] Dahlquist and Björck. *Numerical Methods*. Prentice-Hall, Inc., Englewood Cliffs, 1974.

[Gol90] Herman H. Goldstine. *Remembrance of things past*. ACM Press - Addison-Wesley, Reading, Massachusetts, 1990.

[Gol91] Goldberg. What every computer scientist should know about floating-point arithmetic. *ACM Computing Surveys*, 23(1), March 1991.

[Ham71] Hamming. *Introduction to applied numerical analysis*. McGraw-Hill-Kogakusha, Ltd., Tokyo, 1971.

[JR75] Jensen and Rowland. *Methods of Computation*. Scott, Foresman & Co, Glenview, 1975.

[Lie68] Lieberstein. *A course in numerical analysis*. Harper and Row, New York, 1968.

[Moo63] Moore. *Interval Analysis*. Prentice-Hall, Englewood Cliffs, 1963.

[Mul89] Muller. *Arithmétique des ordinateurs*. Masson, Paris, 1989.

[Wal90] Wallis. *Improving floating-point programming*. Wiley, New York, 1990.

[Wil65] Wilkinson. *Rounding errors in algebraic processes*. Clarendon Press, Oxford, 1965.

[Zac96] Joseph L. Zachary. *Introduction to scientific programming: computational problem solving using Maple and C*. Springer Verlag New York Inc., New York, 1 edition, 1996.

Índice

Impresso em 27 de Fevereiro de 2009 a partir dos seguintes ficheiros:

Analise_Numerica.tex	1.9	2007/04/01
Analise_Numerica.sty	1.10	2009/02/26
Titulo.tex	1.5	2009/02/27
0_Introducao.tex	1.4	2007/04/01
1_Erro.tex	1.8	2009/02/26