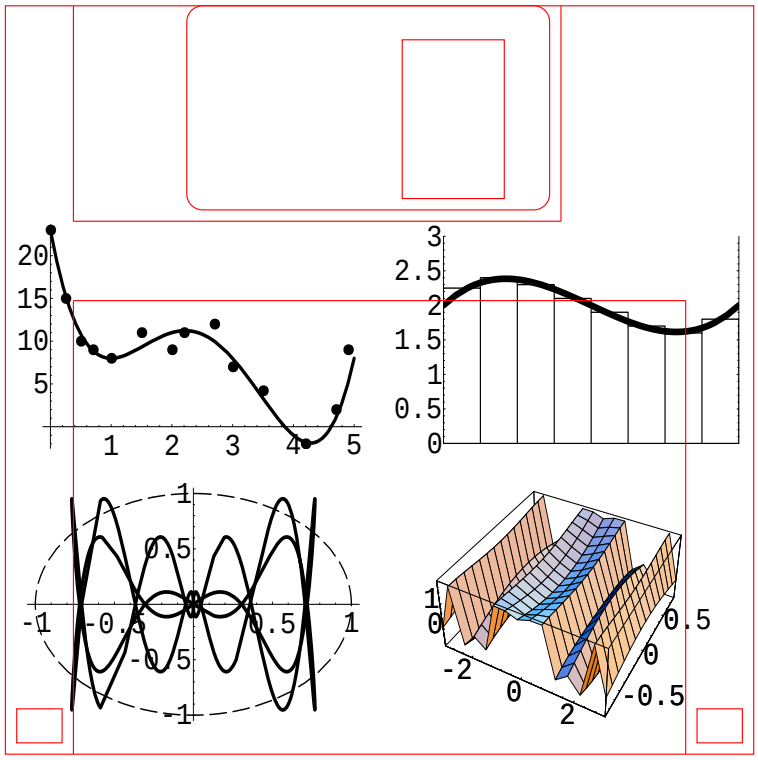
COMPUTAÇÃO NUMÉRICA

Edite Manuela da G.P. Fernandes



$2^{a.}$ **edição**

COMPUTAÇÃO NUMÉRICA

Edite Manuela da G.P. Fernandes

$2^{a.}$ **edição**

com a colaboração de
José Filipe de Sá R. Soares
no desenvolvimento da aplicação

Universidade do Minho, Braga

Título: **Computação Numérica**

$2^{a.}$ edição

Contém disquete com a versão 2.0 da aplicação **CoNum**

Autor: *Edite Manuela da G.P. Fernandes*

Composição: *Texto preparado em* LaTeX *pela autora*

Impressão: *Serviços de Reprografia e Publicações da Universidade do Minho*

Capa: *Edite Manuela da G.P. Fernandes*

TeX é uma marca registada da American Mathematical Society.

ISBN 972-96944-1-9 - $2^{a.}$ edição revista, corrigida e aumentada
(ISBN 972-96944-0-0 - $1^{a.}$ edição)

Depósito Legal $n^{o.}$ 115675/97

Tiragem: 2000 exemplares em Dezembro de 1997

Dedico esta edição às minhas filhas

SOFIA e **JOANA**

e ao meu marido **JOÃO**

Prefácio da segunda edição

Esta obra contém a segunda edição revista, corrigida e aumentada do livro Computação Numérica, editado em 1996, e uma disquete com a aplicação CoNum (versão 2.0).

Este novo prefácio tem como objectivos, em primeiro lugar, listar sucintamente o conteúdo das matérias que foram adicionadas à primeira edição e por último registar mais alguns agradecimentos.

Assim, e começando pelo conteúdo, no Capítulo 4, foi adicionada a subsecção 4.2.4 sobre o método de Newton baseado nas diferenças finitas para a resolução de sistemas de equações não lineares.

No Capítulo 7, as novas subsecções 7.2.7 e 7.2.8 tratam das propriedades estatísticas da solução (modelação) e da aproximação de funções periódicas pela técnica dos mínimos quadrados.

O Capítulo 11 é novo e trata de equações diferenciais com derivadas parciais. É feita uma introdução aos métodos baseados em diferenças finitas para a resolução de equações diferenciais parabólicas, hiperbólicas e elípticas. Nalguns casos, também se apresentam análises de convergência e estabilidade dos métodos.

Em quase todos os Capítulos foram adicionados novos enunciados de problemas não resolvidos.

Nesta segunda parte, gostaria de expressar os meus mais sinceros agradecimentos a todos os funcionários dos Serviços de Reprografia e Publicações da Universidade do Minho que de uma forma directa ou indirecta estiveram envolvidos na edição desta obra (primeira e segunda edições) pela simpatia e o empenho demonstrados. Os meus parabéns pela rapidez com que executaram a tarefa e pela qualidade do trabalho final.

Para terminar, dirijo um agradecimento muito especial à Escola de Engenharia da Universidade do Minho pelo apoio financeiro concedido à edição da obra Computação Numérica.

Braga, Setembro de 1997 *Edite Manuela da G.P. Fernandes*

Prefácio

Objectivos

Este projecto foi iniciado no ano lectivo de 1993/94 e teve o apoio da Universidade do Minho através da concessão de uma licença sabática. A principal motivação para a sua realização deveu-se à lacuna existente no mercado de livros em língua portuguesa, sobre a Computação Numérica, que pudessem ser utilizados como textos de apoio teórico e prático na leccionação de disciplinas neste âmbito.

A Computação Numérica tem como fim a resolução numérica de problemas matemáticos. Os meios utilizados para alcançar este fim são os métodos numéricos, os correspondentes algoritmos e a sua implementação prática no computador.

Na base destas 'ferramentas' está a Análise Numérica. A definição de Análise Numérica tornou-se, hoje em dia, uma questão filosófica. Os termos aproximação, precisão, erros, arredondamentos e aritmética do computador têm sido usados no passado para a definir. No entanto, actualmente, já não são suficientes uma vez que caracterizam uma pequena parte daquilo que é estudado em Análise Numérica. Assim, uma definição[1] mais recente é:

> *Análise Numérica é o estudo dos algoritmos para a resolução de problemas matemáticos contínuos*[2].

Como se verifica, a essência do trabalho desenvolvido pela Análise Numérica é o algoritmo. É claro que, relacionados com o algoritmo estão a identificação e o estudo da propagação dos erros de arredondamento. Além disso, para a maior parte dos problemas matemáticos não é possível calcular soluções exactas. A Análise Numérica calcula aproximações a quantidades desconhecidas. O objectivo da Análise Numérica é desenvolver algoritmos que calculem aproximações usando processos rapidamente convergentes. A este tipo de algoritmos estão associados outros erros, como os de truncatura e de discretização. Na definição acima indicada não é feita referência directa à implementação desses algoritmos no computador. Fiabilidade, eficiência e flexibilidade são objectivos a atingir nesta fase, entre outros.

A Computação Numérica usa a Análise Numérica e o computador para desenvolver métodos numéricos e os correspondentes algoritmos, codificando-os em seguida para terminar

[1] L. N. Trefethen, The definition of Numerical Analysis, SIAM NEWS, Novembro de 1992.

[2] O termo contínuo significa que estão envolvidas variáveis reais e complexas.

na fase de implementação no computador e análise dos resultados. O livro desenvolve-se ao longo destas três vertentes. A primeira, é coberta pela apresentação dos métodos numéricos, seleccionados como mais eficientes e estáveis na resolução de um conjunto muito variado de problemas matemáticos. A segunda vertente é conseguida pela descrição algorítmica e minuciosa dos métodos apresentados. Ao todo, são sessenta e um algoritmos, distribuídos por nove capítulos. Em cada capítulo existe uma secção com o índice dos algoritmos aí desenvolvidos e uma descrição sucinta do que cada um faz. Por fim, a vertente que consiste na implementação dos algoritmos é atingida através da sua codificação em C^{++}. A aplicação resultante funciona em ambiente Windows e está presente na disquete que acompanha o livro. Vários exemplos são fornecidos, em cada capítulo, para exemplificar os resultados obtidos pelas rotinas. Ao todo, são cinquenta e duas rotinas numéricas. A partir do índice é possível identificar o nome de cada rotina e o método que lhe corresponde. A designação de cada rotina é composta por seis letras (excepto num caso) e está relacionada com o método a que corresponde a implementação. É minha convicção de que este livro encorajará o estudante a utilizar os computadores na resolução numérica de uma grande variedade de problemas matemáticos.

A matéria coberta pelo livro pode ser leccionada a alunos das Licenciaturas em Engenharia e Ciência, em disciplinas do âmbito dos *Métodos Numéricos* e corresponde, no mínimo, a duas disciplinas semestrais. Há, assim, alternativas para a definição do conteúdo programático das disciplinas semestrais. Os assuntos foram distribuídos por dez Capítulos:

1. Erros e estabilidade
2. Solução de uma equação não linear
3. Sistemas de equações lineares
4. Sistemas de equações não lineares
5. Valores e vectores próprios de matrizes
6. Interpolação polinomial
7. Aproximação dos mínimos quadrados
8. Integração numérica
9. Equações diferenciais ordinárias
10. Optimização não linear sem restrições

Tendo como necessidade a satisfação das três vertentes acima referidas, durante a escrita deste livro foram ainda delineados os seguintes objectivos:

- a selecção da matéria, a sua organização, a apresentação, o conteúdo e o detalhe deveriam originar um elemento de estudo razoavelmente completo, que fosse facilmente compreendido por estudantes das Licenciaturas em Engenharia e Ciência, com alguns conhecimentos básicos de Matemática e Programação de Computadores;

- o desenvolvimento dos algoritmos deveria permitir aos estudantes a implementação rápida dos métodos, usando qualquer das linguagens que tivessem aprendido. O detalhe com que os algoritmos fossem apresentados, deveria responder a um conjunto de questões normalmente levantadas pelo estudante, nomeadamente as que dizem respeito à paragem dos processos iterativos, à atribuição de valores a parâmetros de entrada e até a falhas de programação;

- a inclusão da disquete permitiria um acesso fácil e rápido, quer para os estudantes quer mesmo para cientistas e engenheiros, a rotinas numéricas para a resolução de problemas matemáticos de complexidade e dimensão médias;

Organização e conteúdo

Em termos gerais, a dependência entre os capítulos é a apresentada na figura 0.1. O Capítulo 1 dos Erros e estabilidade é uma introdução imprescindível à matéria, sem a qual não fará sentido a apresentação e a selecção feita dos métodos descritos. No Capítulo 2 são apresentados alguns métodos para o cálculo de raízes reais e complexas de uma equação não linear. É também analisada a sua convergência e aplicabilidade. A resolução de um sistema de equações lineares, bem como a determinação da inversa de uma matriz e o cálculo do seu determinante são assuntos que podem ser estudados no Capítulo 3. A resolução de um sistema de equações não lineares é tratada no Capítulo 4, que pode ser considerado como uma extensão do Capítulo 2, mas que necessita de alguns conhecimentos do terceiro Capítulo. O cálculo dos valores e vectores próprios de matrizes, quer simétricas quer não simétricas, é abordado no Capítulo 5. Os Capítulos 3 e 5 constituem aquilo que é costume designar por Álgebra Linear Numérica. Os dois capítulos seguintes estudam o problema da aproximação de funções, embora com objectivos e técnicas diferentes. O Capítulo 6 descreve técnicas de interpolação, incluindo as 'splines', e o Capítulo 7 aborda o problema do ajuste de modelos polinomiais, não polinomiais e também os não lineares nos parâmetros, a um conjunto de dados experimentais. O problema da integração numérica de funções é tratado no Capítulo 8. O Capítulo 9 é reservado para a resolução de problemas com equações diferenciais ordinárias com condições iniciais, sem esquecer também os problemas com condições de fronteira. Conhecimentos sobre a matéria leccionada nos Capítulos 3 e 6 são essenciais para a sua compreensão. Finalmente o Capítulo 10 sobre Optimização não linear, sem restrições nas variáveis, trata de um problema que pode ser considerado como directamente relacionado com os dos Capítulos 2 e 4, mas não equivalente, como se verá na devida altura.

Faz-se em seguida uma apresentação mais pormenorizada do conteúdo dos capítulos, ao mesmo tempo que se define um programa para uma disciplina semestral e introdutória no âmbito dos Métodos Numéricos, habitualmente designada por Análise Numérica, Métodos Numéricos I ou Métodos Computacionais I.

Na essência são os Capítulos 1, 2, 3, 6 e 8 que cobrem as matéria consideradas básicas. Fazendo uma apreciação mais específica, é possível introduzir algumas restrições neste conteúdo. Veja-se a figura 0.2.

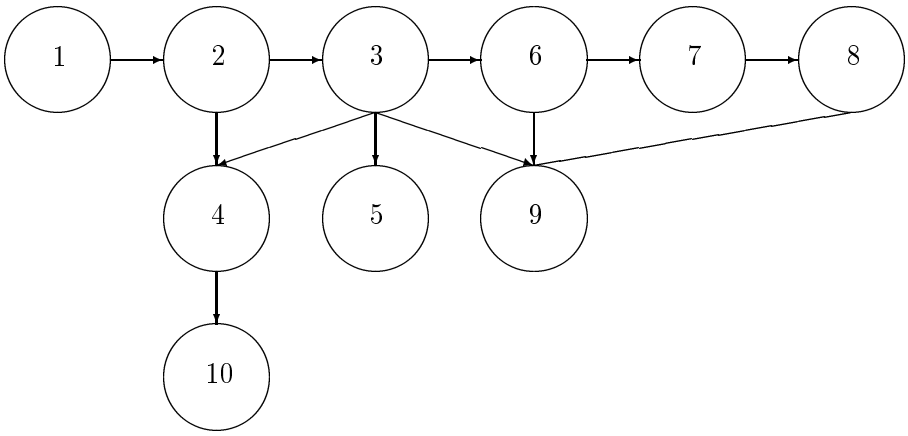


Figura 1: Esquema da dependência dos dez capítulos

No Capítulo 1 estudam-se os erros de arredondamento e sua propagação (em 1.3, 1.4 e 1.5) e os de truncatura (em 1.6) cuja identificação e análise se consideram necessárias em computação numérica. Uma consciencialização das etapas a seguir no desenvolvimento de um programa de computador é feita em 1.2.

Para a resolução de uma equação não linear, o Capítulo 2 introduz processos para a localização das raízes e descreve métodos para as calcular. Em 2.5, 2.7, 2.8 e 2.10 são introduzidos os métodos mais conhecidos: bissecção, secante, Newton-Raphson e ponto fixo. O condicionamento das raízes de polinómios (em 2.2.2) e um critério de paragem de processos iterativos (veja-se em 2.3) são temas de grande importância. Os métodos da falsa posição, em 2.6, e de Laguerre, em 2.9, podem ser deixados para outra altura ou sugeridos para estudo individual e complementar.

No Capítulo 3 e para a resolução de um sistema de equações lineares, é imprescindível falar do seu condicionamento (em 3.2), introduzir os métodos directos, em 3.4, os métodos iterativos, em 3.6, bem como a decomposição da matriz dos coeficientes para o cálculo do seu determinante (veja-se em 3.5). Poder-se-á deixar de fora, numa abordagem mais elementar, a factorização Q R, em 3.5.5, e um processo de redução dos erros de arredondamento, em 3.3.

No domínio da aproximação de funções, o Capítulo 6 cobre a interpolação baseada em diferenças finitas e a interpolação segmentada. Pela sua aplicação mais geral, a interpolação baseada em pontos desigualmente espaçados, em 6.3, torna-se muito comum. Pelas vantagens da sua aplicação, também devem ser introduzidas as funções 'splines', em 6.4. A interpolação com espaçamento constante, em 6.2, pode ser deixada para outra altura mais conveniente.

Do Capítulo 8, sobre integração numérica, as fórmulas de Newton-Cotes, bem como as correspondentes fórmulas compostas tornam-se essenciais, em 8.2. Em 8.4, é possível introduzir algumas das fórmulas Gaussianas, adequadas à resolução de alguns integrais

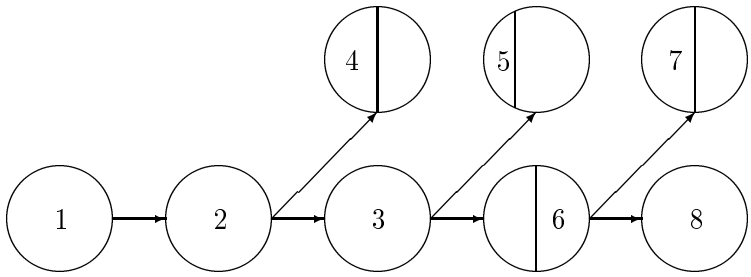


Figura 2: Sequência de capítulos de uma disciplina semestral introdutória

impróprios. A integração de Romberg, em 8.3, pode ser considerada matéria mais complementar.

Além da matéria já identificada, nestes cinco capítulos, como introdutória, também se considera que a resolução de um sistema de equações não lineares pelo método de Newton, em 4.2 do Capítulo 4, os métodos iterativos para o cálculo dos valores, e vectores próprios associados, de maior e menor módulo de uma matriz, em 5.3, bem como o seu condicionamento (em 5.2), do Capítulo 5 e o ajuste de funções pela técnica dos mínimos quadrados (em 7.2) do Capítulo 7, onde são introduzidos os polinómios ortogonais, devem ser considerados assuntos de interesse introdutório, especialmente se esta for a única disciplina leccionada no domínio dos métodos numéricos.

O método da secante, em 4.3 do Capítulo 4, o cálculo do conjunto inteiro dos valores próprios e correspondentes vectores de uma matriz simétrica, em 5.4, e de uma matriz não simétrica, em 5.5, do Capítulo 5 e o problema dos mínimos quadrados não linear, em 7.3, do Capítulo 7, constituem matéria complementar que pode ser deixada para uma segunda disciplina semestral.

Embora o conteúdo programático aqui proposto de uma maneira sucinta resulte da experiência adquirida pela autora, ao longo de mais de dez anos, na Universidade do Minho e com estudantes das Licenciaturas em Engenharia, e partindo do pressuposto de que a diversidade de problemas tratados é importante, especialmente no que diz respeito a estudantes com apenas uma disciplina semestral no seu curriculo, pode aceitar-se um programa que não inclua as referências feitas aos Capítulos 4, 5 e 7, leccionando-se, em vez disto, toda a matéria que engloba os Capítulos 1, 2, 3, 6 e 8.

Duas propostas alternativas são feitas para uma segunda disciplina semestral no âmbito dos métodos numéricos. Esta disciplina chama-se normalmente Métodos Numéricos II, Complementos de Análise Numérica ou Métodos Computacionais II. A primeira sugestão tem sido implementada na Universidade do Minho em três Licenciaturas em Engenharia. Veja-se a figura 0.3.

Na sequência da primeira disciplina e tal como foi sugerido atrás, o método da secante, baseado nas equações de Broyden, (em 4.3) para a resolução de um sistema de equações não lineares surge no Capítulo 4. Este tipo de método tem o seu correspondente nos problemas de optimização do Capítulo 10.

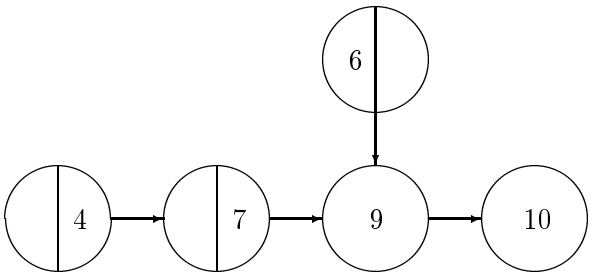


Figura 3: Sequência de capítulos para uma segunda disciplina semestral

Ainda relacionado com o problema tratado no Capítulo 10, sem dúvida um dos temas centrais desta segunda parte, está o problema de mínimos quadrados não linear, que aparece em 7.3, no Capítulo 7.

A primeira parte do Capítulo 6, interpolação polinomial baseada em pontos com espaçamento constante, veja-se em 6.2, é de alguma importância para o Capítulo 9, embora possa ser abordada de uma maneira mais superficial do que a descrita.

A resolução numérica de equações diferenciais é discutida no Capítulo 9. Problemas de primeira ordem e com condições iniciais são tratados em 9.2, usando métodos de passo único e métodos de passo múltiplo. Em 9.3, analisa-se a resolução de um sistema de equações e em 9.4, aborda-se a resolução de equações de ordem igual ou superior a dois. O caso particular das equações 'stiff' é tratado em 9.5 e em 9.6 surgem os problemas com condições de fronteira.

Finalmente no Capítulo 10 é analisada a resolução de um problema de optimização a uma dimensão, onde são apresentados dois tipos de métodos, os de procura e os de aproximação. Veja-se em 10.2. Em 10.3 consideram-se problemas mais gerais, os multidimensionais. Dos métodos mais eficientes incluem-se os métodos de procura de 10.3.3 e 10.3.5 e os métodos baseados nos gradientes, tais como o das direcções de descida máxima, do tipo Quasi-Newton e dos gradientes conjugados, respectivamente em 10.3.8, 10.3.12 e 10.3.14. O método de Newton de segunda ordem é analisado em 10.3.10.

Na figura 0.4 apresenta-se uma alternativa ao conteúdo programático agora descrito. O Capítulo 10 é substituído pelo Capítulo 5. A ênfase, neste caso, pende para a Álgebra Linear Numérica. Se os métodos iterativos, para o cálculo dos valores próprios de maior e menor módulo e vectores associados de uma matriz, foram já introduzidos na disciplina introdutória, então o Capítulo 5 será completado com os métodos baseados em transformações semelhantes. Assim, em 5.4, calcula-se o conjunto inteiro dos valores e vectores próprios de uma matriz simétrica, usando transformações semelhantes, em 5.4.2 e 5.4.4, as propriedades das sequências de Stürm, em 5.4.6, e o método da potência para os vectores (5.4.8). Para as matrizes não simétricas usam-se transformações elementares, em 5.5.2, o método Q-R, em 5.5.4, e a versão para vectores complexos do método da potência (5.5.6).

Comentários sobre o 'software'

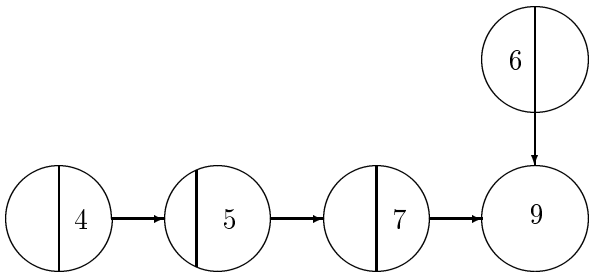


Figura 4: Sequência de capítulos para uma segunda disciplina semestral alternativa

Os algoritmos foram implementados e codificados na linguagem C^{++} para Windows 3.1. Os resultados das experiências computacionais realizadas, e que são apresentados ao longo do livro, foram obtidos usando um Computador Pessoal 486 com coprocessador aritmético, com quinze dígitos de precisão para números no formato de vírgula flutuante de dupla precisão.

A disquete que acompanha o livro define uma aplicação em ambiente Windows e contém cinquenta e duas rotinas numéricas que correspondem a algoritmos apresentados ao longo dos últimos nove Capítulos.

O utilizador deve estar alertado para o facto de que os resultados obtidos por uma rotina podem variar, dependendo do computador utilizado. Embora estas diferenças sejam muito pequenas, podem, por vezes, ser significativas se os cálculos forem sensíveis aos erros de arredondamento que se cometem ao longo do processo numérico.

Agradecimentos

Os primeiros agradecimentos são dirigidos ao *Engº* José Filipe de Sá R. Soares que codificou os sessenta e um algoritmos numéricos, apresentados no livro, em C^{++} e desenvolveu a aplicação em ambiente Windows. A fase de teste das cinquenta e duas rotinas numéricas exigiu um esforço adicional e muita paciência devido ao tipo de trabalho repetitivo e pouco estimulante.

Agradecimentos são também devidos aos *Engºs* Maria Teresa T. Monteiro e Gaspar José B.Q.A. Machado, colegas do Departamento de Produção e Sistemas e ao Dr. João Honório Matias do Departamento de Matemática da Universidade de Trás-os-Montes e Alto Douro pela colaboração prestada na realização dos gráficos apresentados nos Capítulos 1, 2, 9 e 10 do livro.

Questões legais

Embora as rotinas tenham sido testadas com um número razoável e variado de problemas, não é possível garantir precisão e funcionamento perfeito das rotinas para todos os problemas. A aplicação não deve ser usada por forma a que resultados incorrectos na resolução de um problema possam vir a originar prejuízos pessoais ou perdas de propriedade. A sua utilização neste sentido é da inteira responsabilidade do utilizador.

Foram feitas referências a várias marcas registadas: C^{++} refere-se a um 'software'

com direitos de autor da Borland International, Inc. e Windows refere-se também a um 'software' com direitos de autor da Microsoft Corporation.

Braga, Maio de 1996 *Edite Manuela da G.P. Fernandes*

Conteúdo

Capítulo 1

Erros e estabilidade

1.1 Introdução

A *Análise Numérica*, tal como é conhecida hoje em dia, é posterior à II Guerra Mundial. No entanto, as suas ideias básicas e os seus objectivos têm sido usados por muitos matemáticos, filósofos e astrónomos para desenvolver fórmulas matemáticas e tabelas, ao longo dos séculos. Dos nomes mais conhecidos podemos citar Pitágoras (580 - 500 a.C.), Arquimedes (287 - 212 a.C.) e Euclides. Mais recentes temos, Napier e Bürgi do século XVI, Pascal, Leibniz, Briggs, Kepler, Descartes, Newton, Galileu e Fermat do século XVII. Napier e Bürgi foram os primeiros a introduzir os logaritmos, tendo marcado uma época muito importante no desenvolvimento da Matemática. O estudo dos logaritmos pode ser considerado como um percursor da Análise moderna. Briggs (1556 - 1630) introduziu grandes avanços na construção de tabelas de logaritmos e as suas ideias originais e revolucionárias contribuíram para o aparecimento da Análise Numérica. Foi, logo após o aparecimento dos computadores electrónicos, nos anos quarenta, que os cálculos automáticos de grande escala se tornaram uma ferramenta importante nos campos da Ciência e da Tecnologia. Por exemplo, os cálculos que Delaunay realizou durante 20 anos, no século XIX, levou, cem anos mais tarde, 20 horas de computação automática. Um dos campos onde se sentia uma maior necessidade de invenção de máquinas de calcular era o da Astronomia, devido ao volume de operações a efectuar.

A Análise Numérica é um campo muito especializado que requer conhecimentos de Matemática. Este campo está directamente relacionado com a resolução de problemas matemáticos, tendo como ferramenta principal os computadores[1] electrónicos.

[1]Os computadores têm a sua origem nos ábacos. Sabe-se que os ábacos eram usados na altura do Império Grego. O ábaco asiático apareceu pela primeira vez na China e foi exportado para o Japão no século 16. Nestes países ainda hoje se usa o ábaco. O inglês E. Gunter (1581-1626) desenvolveu, em 1624, a primeira régua de cálculo ainda hoje usada por alguns engenheiros. O alemão W. Schickard que viveu entre 1592 e 1635 é considerado o pai da calculadora mecânica. B. Pascal (1623-1662), um matemático francês, desenvolveu, aos 18 anos, uma máquina que adicionava e subtraía. A multiplicação tornava-se muito lenta pois era feita por sucessivas adições. Trinta anos mais tarde, o filósofo e matemático G.W. Leibniz (1646-1716) construíu uma máquina que executava as quatro operações. Ainda no século XVIII, P. M. Hahn

Os dois aspectos principais a considerar na computação automática de soluções de problemas matemáticos são:

1. o desenvolvimento e a análise de métodos computacionais para a resolução dos problemas, e

2. a implementação destes métodos com o objectivo de executar cálculos científicos.

A estes métodos chamaremos *Métodos Numéricos*. Envolvem procedimentos e fórmulas que nos levam à aproximação de um problema matemático por um problema numérico ou à resolução de um problema numérico. A *Análise Numérica* é pois o campo que estuda o comportamento dos Métodos Numéricos. Este estudo engloba a existência e unicidade das soluções, a convergência, a estabilidade, a eficiência, a complexidade, etc.

O desenvolvimento de *algoritmos numéricos* a partir dos métodos numéricos é feito tendo em atenção o tipo e as características do problema e o contexto em que surge. O algoritmo deve conter todos os detalhes computacionais, nomeadamente em termos das condições iniciais e de paragem do procedimento. O conceito de Método Numérico é mais geral do que o de algoritmo numérico.

Uma das áreas onde os computadores, quer os microcomputadores pessoais quer os de grande porte, têm sido mais usados, desde o início do seu aparecimento, é a da *análise* e *solução numéricas* de uma enorme variedade de problemas matemáticos. O seu uso tem permitido ao utilizador obter soluções numéricas num período de tempo aceitável. Os computadores de grande porte continuam a ser as ferramentas cruciais na resolução de problemas muito complexos e de grandes dimensões, apesar do rápido desenvolvimento e do uso generalizado dos microcomputadores. A maior parte dos utilizadores continua, hoje em dia, a preferir a utilização dos microcomputadores para a resolução dos seus problemas numéricos, desde que a capacidade de memória assim o permita e a precisão e a velocidade de computação não sejam consideradas factores críticos na resolução. Esta

construíu uma dúzia de máquinas, mas foi só no século XIX que as calculadoras mecânicas começaram a ser construídas em grande número. C.X. Thomas (1785-1870) construíu uma máquina que pela primeira vez resolvia o problema do "transporte"exactamente. O americano W.S. Burroughs desenvolveu, em 1884, a primeira calculadora de somar com impressão e teclas. O inglês C. Babbage (1791-1871) desenvolveu uma máquina que podia ser controlada por um conjunto de instruções, nos mesmos moldes dos computadores modernos. Em 1833 Babbage começou a desenvolver uma máquina que continha uma unidade de memória, uma unidade aritmética, uma unidade de controlo baseada em cartões perfurados e unidades de entrada e saída de informação. No entanto, foi H. Hollerith, que viveu entre 1860 e 1929, que desenvolveu um método para armazenar informação em cartões perfurados. O desenvolvimento dos computadores modernos começou em Berlim, em 1934/35, com K. Zuse, que contruíu a sua primeira máquina, a Z1. Usou o sistema binário e todas as quatro operações aritméticas eram realizadas pelas operações lógicas, "e", "ou"e "não". Outras máquinas se seguiram e em 1945 surgiu a Z4. Também nos E.U.A. começaram a aparecer os primeiros computadores. H.H. Aiken (1900-1973) construíu o seu primeiro modelo MARK I em 1944. Em 1946 surgiu o primeiro computador totalmente electrónico, construído por J.P. Eckert e J.W. Mauchly na Universidade de Pennsylvania, com o objectivo de resolver um problema de equações diferenciais por métodos iterativos. Também em Inglaterra (1943) foi construído um computador a partir das ideias de A.M. Turing (1912-1954) e que utilizava aritmética binária. Um matemático do nosso século, cujas ideias muito contribuíram para o desenvolvimento dos computadores modernos foi J.V. Newmann (1903-1957) (Hämmerlin e Hoffmann (1991)).

preferência deve-se essencialmente a dois factores: um, está relacionado com a facilidade de utilização do sistema operativo[2] MS-DOS, e o outro, com a diminuição dos preços de venda ao público. Outra característica dos microcomputadores pessoais, que os tornam tão manuseáveis, resulta da possibilidade de qualquer utilizador poder individualmente usá-lo sem necessidade de recorrer a uma distribuição de recursos computacionais, que depende dum administrador do sistema.

Compete aos analistas numéricos desenvolver algoritmos numéricos para a resolução numérica de problemas matemáticos, nos quais, os efeitos da aritmética discreta dos computadores permaneçam pouco significativos, mesmo quando o número de operações a efectuar é muito elevado.

Alguns dos algoritmos desenvolvidos para resolver problemas matemáticos padrão encontram-se implementados e incorporados em bibliotecas de programas de computação numérica, não sendo precisos, para a sua utilização, conhecimentos muito profundos de Análise Numérica. No entanto, e porque para um certo tipo de problemas, podem existir vários algoritmos, torna-se conveniente saber seleccionar o algoritmo mais adequado para a resolução do problema entre mãos, tendo em conta as suas especificações. Para um utilizador casual do 'software' numérico é suficiente ter uma ideia *intuitiva* e *qualitativa* dos erros que se cometem ao longo do processo numérico.

Nos casos em que o problema matemático tenha certas particularidades que o distinguem dos outros, pode não existir, nas bibliotecas numéricas, um programa que calcule a solução desse problema, sendo necessário ao utilizador desenvolver o seu próprio algoritmo/'software'. Nesta situação, o utilizador necessita de fazer um estudo mais pormenorizado sobre a aplicabilidade e estabilidade das técnicas numéricas. Para o analista numérico, que desenvolve 'software' numérico, uma compreensão meramente intuitiva não é suficiente. Ele deve ser capaz de prever *quantitativamente* como um dado algoritmo se comportará numericamente e explicar teoricamente porque razão algoritmos diferentes para obter a mesma solução se comportam de maneiras diferentes.

Nas duas situações apresentadas, o utilizador deve possuir um mínimo de conhecimentos no domínio da Análise Numérica.

Assim, este trabalho tem como objectivo principal dar a conhecer métodos numéricos que possam, por razões de estabilidade, convergência, fiabilidade e utilização fácil, ser implementados para determinar soluções práticas de problemas de engenharia, com a utilização de computadores.

A utilização de computadores na resolução de problemas numéricos origina o aparecimento de um certo tipo de erros, que não surgiriam se fosse possível usar aritmética exacta.

[2]Sistema operativo é um conjunto de instruções de programas necessário para gerir e coordenar as diversas partes de um sistema computacional. O sistema operativo age como um intermediário entre o utilizador - ou os programas do utilizador - e o 'hardware' do sistema e unidades periféricas. Os sistemas operativos para os microcomputadores chamam-se DOS (Disk Operating Systems). Resumidamente, podemos dizer que o sistema operativo é responsável pela tradução dos comandos usados pelo utilizador nos comandos do 'hardware'. Um sistema como o MS-DOS (PC-DOS ou UNIX), baseado em comandos, permite que o utilizador introduza os comandos para um monitor. Em 1981 apareceu a primeira versão do MS-DOS para computadores pessoais. Presentemente temos já as versões 6.

A *presença*, a *acumulação* e a *propagação* desses erros, podem, em certos casos, influenciar bastante os resultados finais. O resultado calculado pode não ter nada a ver com a solução exacta do problema.

1.2 Etapas no desenvolvimento de 'software' numérico

1.2.1 Modelação matemática

Para investigar os problemas que surgem nas áreas das Ciências, Engenharia e outras, torna-se imprescindível a construção de modelos matemáticos que descrevem aproximadamente o comportamento de um sistema real.

Embora os sistemas reais variem em complexidade e nós não tenhamos controlo sobre eles, podemos, no entanto, controlar a complexidade dos modelos matemáticos que construímos para descrever o comportamento dos sistemas reais. Sistemas complexos podem vir a ser aproximados por modelos matemáticos simples e as equações associadas podem ser resolvidas analiticamente. No entanto, estes modelos mais simples muito provavelmente não dão uma descrição exacta do comportamento da situação real. Uma descrição mais precisa do sistema real pode ser conseguida incorporando, no modelo matemático, mais características do sistema real. As equações resultantes tornar-se-ão mais difíceis de resolver. Nestes casos, não existirão provavelmente soluções analíticas e teremos, como alternativa, de recorrer a soluções aproximadas obtidas através dos *métodos numéricos*.

Têm sido propostos muitos diagramas para descrever o processo de modelação matemática (Blum, Niss e Huntley (1989)). Um exemplo simples de um diagrama que descreve o processo de modelação matemática no âmbito de um cenário científico é o que está representado na figura 1.1.

Na construção de um modelo matemático, tenta-se seleccionar um, dentre os modelos já conhecidos e habituais, introduzindo as variáveis do modelo, recolhendo e apresentando os dados. Temos, assim, a etapa 1 do processo. Nesta etapa deve existir grande flexibilidade por forma a permitir diversas abordagens e alguma criatividade individual. Em seguida, passa-se à etapa 2, na qual se define o problema matemático através da análise matemática das equações encontradas. Na etapa 3 o problema é resolvido e os resultados obtidos são interpretados e validados na etapa 4. A realização prática das duas últimas etapas está fortemente relacionada com o problema real. Das observações feitas deste problema matemático é possível verificar a proximidade do comportamento do modelo construído em relação à situação real.

1.2.2 Selecção do algoritmo

Em Computação Científica o principal objecto de estudo é o algoritmo. Em Análise Numérica, os algoritmos são um meio para atingir um fim e que é a solução do problema matemático. Não é, contudo, esta a visão das Ciências da Computação.

O diagrama de um modelo simples abstracto que descreve implicitamente as etapas no processo de selecção de um algoritmo é o que está representado na figura 1.2 (veja-se

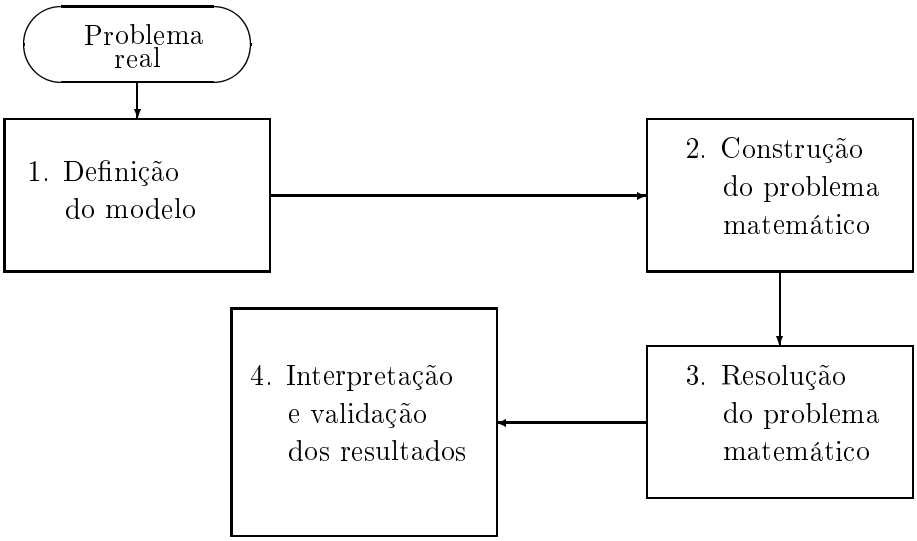


Figura 1.1: Diagrama do processo de modelação

também em Rice (1979)).

A função selecção $S(x)$ faz corresponder a cada problema do espaço X um algoritmo do espaço A dos algoritmos. Associada ao problema e ao algoritmo seleccionado existe a função desempenho, $d(A, X)$. Esta função tem no espaço imagem d uma grandeza que a mede. Assim o resultado dessa medição é $\|d\|$, que fornece o *desempenho* do algoritmo seleccionado para a resolução daquele problema específico.

O objectivo final deste processo é ter a função selecção $S(x)$ a dar um desempenho com valores elevados. Associado ao modelo existem várias questões relacionadas com a 'melhor' função selecção $S(x)$. A 'melhor' selecção é pois aquela que dá o máximo desempenho para cada problema.

1.2.3 Teoria dos algoritmos óptimos

Eram três as etapas mais importantes no desenvolvimento da análise algorítmica (Traub (1974)):

1. síntese de *um* algoritmo;
2. análise de *um* algoritmo;
3. análise de *uma classe* de algoritmos.

No passado a ênfase era dada à síntese de um algoritmo. Em 1947 teve início a segunda etapa com uma análise muito cuidada de alguns algoritmos. Nos anos sessenta e setenta

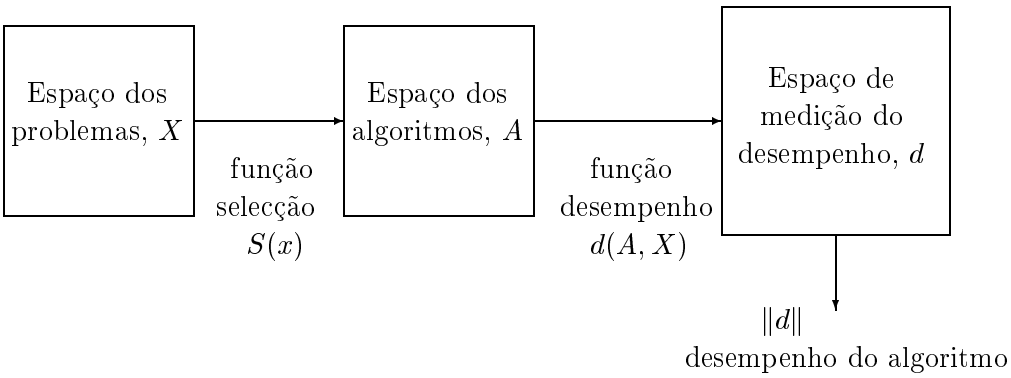


Figura 1.2: Diagrama de um modelo de selecção de algoritmos

começou-se a dar mais atenção às classes de algoritmos numa tentativa de encontrar o melhor deles. Este tema tem sido objecto de desenvolvimentos significativos e existe, hoje em dia, uma comunidade de especialistas fortemente interessada na designada *complexidade computacional*.

As razões mais importantes que levam as pessoas a estudar a complexidade computacional dos algoritmos podem ser enumeradas:

1. necessidade de construir 'bons' e novos algoritmos;

2. necessidade de identificar os 'maus' algoritmos;

3. necessidade de criar uma teoria dos algoritmos que estabeleça limites teóricos de computação.

Uma teoria da complexidade adequada para a Análise Numérica deveria possuir, no mínimo, teoremas que estabeleçam o tempo de computação dos algoritmos básicos, quer para os determinísticos quer para os estocásticos, em termos de números, com a inclusão da precisão desejada e tamanho do problema associado com os dados.

Mesmo quando se usa aritmética exacta, os algoritmos numéricos resolvem os problemas numéricos aproximadamente, com uma precisão, digamos, de ϵ. A teoria da complexidade das Ciências da Computação deve ser modificada de modo a poder tratar estas situações, se pretende ser de alguma utilidade na Análise Numérica. Os especialistas das Ciências da Computação estabelecem que um algoritmo, definido por uma máquina M, é *tratável* (é de tempo polinomial, ou está em P) se o tempo $T(y) = Tempo(y)$ de computação associado às entradas y satisfaz o limite

$$T(y) \leq c\ (tamanho\ y)^q,$$

em que as constantes c e q dependem apenas da máquina. *Tempo* dá o número de operações e *tamanho* dá o número de 'bits' da máquina. Um problema diz-se tratável se existe um algoritmo tratável que o resolva.

A formalização da noção de problema, a interpretação dos conceitos de *Tempo* e *tamanho* das entradas em $\mathbf{R}$, bem como a consideração de fenómenos importantes em Análise Numérica, tais como o condicionamento de problemas e a estabilidade numérica dos algoritmos, levaram Smale (1990) a estender a definição de algoritmo tratável à Análise Numérica. Assim,

$$T(\epsilon, y) \leq c_1 t(y)^{q_1} + c_2 |\log \epsilon|^{q_2} + c_3 (\log W(y))^{q_3}$$

em que as constantes $c_i, q_i, i = 1, 2, 3$ são funções da máquina ou do algoritmo, $W(y)$ é uma função peso e pretende medir a *dificuldade do problema*, através de uma norma ou mesmo do número de condição, ϵ dá a precisão do resultado e $t(y)$ designa *tamanho* das entradas y. O espaço das entradas numa máquina (de dimensão infinita) é o espaço linear $\mathbf{R}^\infty$ de todas as sequências de números reais da forma $y = (y_1, y_2, \ldots, y_n, 0, 0, \ldots)$. Neste caso tem-se $t(y) = n$.

A *complexidade* de um algoritmo está relacionada com o número de operações elementares necessárias para resolver um tipo de problemas. Por exemplo, na resolução de um sistema de n equações lineares em n variáveis, o algoritmo de eliminação de Gauss (Capítulo 3, em 3.4.2) é $\mathcal{O}(n^3)$, ou seja, o número de operações é da ordem de n^3, ou ainda, esse número é proporcional a n^3.

Parece lógico pensar-se que o tempo que o computador leva a resolver o problema está directamente relacionado com o número de operações.

Quando um algoritmo leva um certo tempo a resolver um problema e esse tempo é proporcional a n^k, sendo n a dimensão do problema, então o algoritmo diz-se de *tempo polinomial*. A maior parte dos algoritmos numéricos pertence a esta *classe polinomial*.

Existe também outra classe de algoritmos que são de *tempo exponencial* e para estes o tempo de execução aumenta exponencialmente com o número de operações n, por exemplo: 3^n, n^n e e^n. Os algoritmos desta classe são sempre mais lentos do que os polinomiais, e tanto mais lentos quanto maior for n. Assim, existem duas classes de problemas.

- Um problema é um elemento da classe designada por P se pode ser resolvido por um algoritmo determinístico, isto é, um algoritmo bem definido, no qual, o tempo de execução é proporcional a n^k (de tempo polinomial).

- Um problema é um elemento da classe designada por NP se não pode presentemente ser resolvido por algoritmos determinísticos polinomiais. Problemas desta classe são resolvidos teoricamente por algoritmos do *tipo exponencial*, mas os métodos usados não são eficientes. Este facto não exclui a possibilidade de, num futuro próximo, se encontrar um algoritmo determinístico polinomial.

A classe de problemas NP inclui a classe P como subconjunto. A designação NP significa *não determinístico polinomial* e o termo não determinístico quer dizer que o algoritmo pode conter componentes aleatórias.

Outra classe de problemas, subconjunto da classe NP, é a classe NP *completa*, NPC. São problemas NP de uma certa dificuldade de tal forma que se 'se poder mostrar que algum deles pertence à classe P então todos os membros da NP pertencem também à classe P'.

Embora esta questão da complexidade dos algoritmos pareça ser apenas de interesse teórico, ela tem incentivado, na prática, o desenvolvimento de algoritmos cada vez mais eficientes.

1.2.4 Diferença entre algoritmo e 'software' numéricos

É importante saber distinguir entre algoritmo e 'software' numéricos. Define-se *algoritmo numérico* como sendo um conjunto de directivas para executar operações matemáticas, criadas para obter a solução de um determinado problema matemático. Um algoritmo não fica bem definido se surgirem dúvidas em relação à operação que deve ser executada a seguir. Um algoritmo é pois uma técnica numérica para resolver um problema matemático.

Quando implementado, como uma rotina ou num 'package', o algoritmo torna-se uma peça de '*software*'.

No passado, o algoritmo era visto como um processo que terminava depois de executado um número finito de etapas. No entanto, nem todos os problemas podem ser resolvidos nesse número de etapas. Assim, os algoritmos, para a resolução desse tipo de problemas, necessitariam de uma sequência infinita de operações, embora numerável. É claro que apenas um número finito é realmente executado. Considera-se mesmo que quanto maior é o número de etapas efectuadas maior é a precisão. Os algoritmos numéricos podem ser classificados em duas grandes classes, dependendo da natureza do problema matemático. Este, pode ser *finito* ou *infinito*. Por exemplo, a multiplicação de matrizes, a resolução directa de sistemas de equações lineares e a determinação dos zeros de uma equação do $2^{o.}$ grau são exemplos de problemas finitos. Três exemplos de problemas de natureza infinita: a determinação dos zeros de uma equação transcendente, a resolução de equações diferenciais e o cálculo de certos integrais.

Quando se comparam classes de algoritmos, para determinar o melhor, é habitual usar uma medida do custo do algoritmo. Essa medida pode ser baseada no número total de operações aritméticas executadas, como caracteriza a complexidade. No entanto, no estudo do desempenho dos algoritmos, outras propriedades devem ser incluídas, tais como: a *estabilidade*, o *domínio de convergência* e a *eficiência*.

Existem também duas classes de 'software' numérico. A primeira classe envolve os algoritmos que podem ser implementados como um processo de decisão finito. Isto é, a implementação pode ser descrita por um número finito de regras de decisão, tendo em atenção a aritmética discreta do computador. A outra classe de 'software' envolve algoritmos que computacionalmente não são processos finitos. Isto pode dever-se ao facto do algoritmo numérico não ser finito, como acontece num processo iterativo, ou ao facto de que quando o algoritmo é implementado como uma rotina, a aritmética discreta faz com que ele se torne num processo de decisão infinito.

Quando se desenvolve um algoritmo, é costume apresentar alguns resultados de experiências computacionais realizadas com o 'software' - implementação específica do algoritmo. Na realização destas experiências, o utilizador tem sempre um conjunto de objectivos a atingir. O principal é conhecer o *desempenho* daquela rotina, quando ela é implementada para resolver uma classe específica de problemas. Muitas pessoas pensam na palavra *desempenho* como significado da velocidade de um programa, medida em termos do número de operações ou do tempo gasto na execução num certo computador. Será mais sensato saber se o *desempenho* é bom, tendo também em conta outros critérios de medição, ou mesmo, ver se a rotina produz resultados correctos quando os valores de entrada introduzidos forem válidos. Assim, outros critérios não menos importantes são: a *eficiência*, a *fiabilidade*, a *simplicidade de uso* e o facto de ser *portátil* e *flexível*.

1.2.5 Desenvolvimento de 'software' numérico

Para se chegar à produção de rotinas eficientes, como elementos essenciais de 'software' numérico, é necessário preencher um conjunto de requisitos. Para isso, deve ser seguido um procedimento que engloba as seguintes etapas:

- A primeira etapa refere-se à *formulação* do problema e consiste na definição do modelo matemático, bem como na especificação dos dados do problema, do tipo e quantidade de resultados que se pretendem obter tendo em atenção que os cálculos numéricos vão ser executados num computador.

  Nesta etapa devem estar envolvidos não só os cientistas, ou quem estiver interessado no problema, mas também os matemáticos que verificarão se a formulação encontrada tem solução (e se ela é única) e se está de acordo com as expectativas do autor do problema.

- A segunda etapa envolve a *selecção* do método a usar. Define-se método como sendo a(s) fórmula(s) matemática(s) que deve(m) ser usada(s) para encontrar a solução do modelo matemático. Esta selecção compreende uma procura entre os métodos existentes ou mesmo a criação de algum método, caso se torne necessário.

  Nesta etapa temos de decidir por um método numérico aceitável, isto é, por um método que nos resolva o problema de uma maneira tão económica e precisa quanto possível.

- Uma vez decidido qual o método a usar, deve passar-se à terceira etapa que consiste em fazer a descrição detalhada do processo computacional, gerando o *algoritmo* numérico.

- A última etapa consiste em *converter* o algoritmo num elemento de 'software' numérico. É pois a realização física do algoritmo. Consiste num *programa* de computador e devem ser utilizadas, da melhor maneira possível, todas as capacidades computacionais do sistema.

O utilizador pode usar uma linguagem máquina[3] ou uma linguagem orientada para o problema, de alto nível, mais simples, cujas instruções executáveis estão mais perto da linguagem corrente falada.

Uma grande variedade de linguagens de alto nível têm sido implementadas nos microcomputadores e esse número continua a crescer. O objectivo destas linguagens é o de permitir ao utilizador uma comunicação, tão simples quanto possível, com o computador. As diferentes linguagens têm sido desenvolvidas com o intuito de servir as 'necessidades' particulares dos diferentes utilizadores. Uma das linguagens mais usadas em microcomputadores é o BASIC[4]. As outras são: FORTRAN[5] , Algol, COBOL, Pascal[6], C[7], PILOT, LOGO e outras mais especializadas como a APL, LISP e Prolog. As mais usadas para o cálculo científico, devido às suas características especiais são as linguagens FORTRAN, Algol, BASIC, Pascal, C e APL. A linguagem COBOL é orientada para as aplicações comerciais, ideal para tratar estruturas de uma enorme quantidade de dados numéricos e não numéricos. LISP e Prolog são mais usadas em trabalhos de investigação no domínio da inteligência artificial e manipulação simbólica. A linguagem PILOT permite a preparação eficiente de material educacional e LOGO é uma linguagem gráfica.

Após estas quatro etapas, o programa deve ser testado e a documentação produzida. Esta documentação deve abranger a análise dos erros cometidos no processo, quer de truncatura quer de arredondamento, e as conclusões a retirar dos resultados obtidos.

[3]Desde a II Guerra Mundial até 1954 quase toda a programação era feita em linguagem máquina ou 'assembly'. Mais tarde apareceram as primeiras linguagens de programação semi-automáticas, tais como o Autocode e o Speedcoding, mas não eram nada eficientes. Quando apareceram os primeiros computadores com 'hardware' incorporado para executar aritmética de vírgula flutuante e indexação, surgiu então a necessidade de desenvolver linguagens de programação mais modernas.

[4]A palavra BASIC é a sigla de Beginners All-purpose Symbolic Instrution Code e foi desenvolvida no Colégio de Dartmouth, em 1960, como uma linguagem de programação para a generalidade das aplicações, particularmente para sistemas de tempo repartido.

[5]FORTRAN, que vem de FORmula TRANslator, foi a primeira linguagem de alto nível a ser desenvolvida pela IBM em meados dos anos 50. Sofreu muitas alterações ao longo do tempo. As versões mais conhecidas, FORTRAN IV e FORTRAN 77 têm sido muito usadas pela comunidade científica e pelos engenheiros. O grande investimento que tem sido feito em programas em FORTRAN garante que a linguagem será usada, pelo menos, até ao próximo século. A comissão FORTRAN 8X desenvolveu uma nova versão, o FORTRAN 99, que inclui gráficos, processamento de 'arrays' mais adequado, possibilidade de definir novos tipos de variáveis e uma sintaxe mais moderna.

[6]A linguagem Pascal tem já hoje muitos adeptos. Foi desenvolvida na Suiça, em 1970, como uma ferramenta de ensino. Existem já várias versões do Turbo Pascal, um compilador rápido que apareceu em 1983. A versão 4 data de 1987.

[7]C é uma linguagem de programação estruturada usada pelos profissionais de programação. O código C é portátil e muito flexível, permitindo-nos fazer coisas que não seriam possíveis com outras linguagens. Esta linguagem está muito ligada ao sistema operativo UNIX (que é escrito em C). Existem também várias versões, incluindo a Turbo C. A mais recente Turbo C^{++} data de 1992.

1.2.6 Critérios de desempenho do 'software' numérico

O 'software' numérico desenvolvido, especialmente para uma utilização mais generalizada, deve possuir algumas das características, que a seguir se enumeram (Crowder, Dembo e Mulvey (1979)):

1. *Domínio de aplicação*, que define as restrições, se elas existirem, nos dados do problema. Se os dados não são aceitáveis, o procedimento deve fornecer informação sobre esse facto. A implementação com dados não aceitáveis pode originar:

   - o bloqueio do processo, não se conseguindo obter resultados;
   - a produção de resultados completamente errados, sendo reconhecidos como tal;
   - a produção de resultados que parecem 'bons', sendo por isso considerados correctos pelo utilizador.

   A última situação é, sem dúvida, a mais perigosa.

2. *Diagnóstico.* Caso ocorra uma situação de falha, o utilizador deve ter a possibilidade de inspeccionar um parâmetro cujo valor está relacionado com essa falha.

3. *Simplicidade de uso.* Nem sempre é possível fazer procedimentos que sejam eficientes, flexíveis e simples de usar, uma vez que estas características entram em conflito. Mesmo que os procedimentos sejam para ser usados por utilizadores não especialistas, é importante que certas decisões, durante a implementação, sejam tomadas pelo próprio utilizador, Cox (1974). O utilizador deve interagir com o procedimento tomando decisões ao longo do processo. No dizer de Cox, "a máquina executa o trabalho mecânico sob a supervisão do utilizador".

4. A *fiabilidade* dos procedimentos é influenciada por muitos factores que dependem do problema a resolver. Entre eles está a estabilidade numérica do algoritmo seleccionado.

5. A *documentação produzida* para os procedimentos deve fazer uma descrição do algoritmo, dos seus parâmetros, dos indicadores de erros, do tempo de execução, da quantidade de memória necessária e da precisão atingida.

6. *Portátil.* O procedimento escrito numa linguagem de alto nível poderá ser usado em qualquer máquina que possua um compilador dessa linguagem. Os algoritmos devem ser implementados de modo a sofrer poucas ou nenhumas alterações em máquinas com esse compilador. Como se sabe, existem várias versões do BASIC. As mais recentes, que até permitem, num certo sentido, programação estruturada são: Microsoft Basic, Hewlett Packard BASIC e BBC BASIC. No entanto, esta multiplicidade de versões pode criar problemas se se pretende que um procedimento corra num conjunto variado de computadores, isto é, que ele seja portátil. Se um dos objectivos do procedimento é que ele seja portátil, então, deve ser escrito usando apenas as características aceites na generalidade das versões da linguagem usada.

7. *Flexível.* O procedimento deve ser suficientemente flexível por forma a ser capaz de tratar todos os problemas do mesmo tipo, que se apresentem com particularidades variadas. Por exemplo, se as derivadas de uma função não puderem ser obtidas analiticamente, o procedimento deve incluir uma opção que as calcule numericamente.

8. *Modular.* Os procedimentos devem ser escritos, sempre que for apropriado, de uma maneira modular. Esta estratégia tem como objectivo colocar o procedimento mais compreensível para os utilizadores não experientes e, para os mais experientes, dá-lhes a possibilidade de modificá-lo e estendê-lo mais facilmente por módulos.

9. *Eficiente.* Têm sido feitas tentativas para tornar a codificação do procedimento eficiente e reduzir o seu espaço de trabalho. Uma tentativa de maximizar a eficiência entrará em conflito com a característica da modularidade. Uma comparação possível da eficiência pode ser feita a nível do número aproximado de operações a executar e espaço de trabalho. No entanto, quando se usa uma das linguagens de alto nível, FORTRAN ou Algol, uma contagem das operações de multiplicação/divisão não é significativa a nível do tempo de execução. De facto, o tempo de execução dos ciclos-DO e de acesso a variáveis em ficheiros tendem a dominar.

10. *Extensível.* Tem-se pensado muito na possibilidade da escolha de parâmetros de entrada e de saída dos procedimentos por parte do utilizador. Os procedimentos mais recentes e avançados dão a arbitrariedade desta escolha, uma vez que deve ser o utilizador a tomar certas decisões cruciais. A máquina só realiza os aspectos puramente matemáticos do processo.

1.2.7 Divulgação de resultados computacionais

As experiências computacionais são a única abordagem viável para a determinação do desempenho de 'software' numérico. Na realização dessas experiências devem-se usar os princípios básicos da *experimentação científica*. Estes princípios consistem na:

- formulação de hipóteses e enumeração explícita de conjecturas;
- definição da medida e dos critérios objectivos para a medição do desempenho;
- descrição completa do *ambiente* experimental;
- reprodução facilitada de resultados;
- publicação dos resultados experimentais obtidos.

Para a publicação dos resultados de uma experiência, deve-se fazer uma descrição sumária e organizada da experiência através do uso de quantidades numéricas que descrevam razoavelmente bem o comportamento do conjunto de resultados obtidos. Por exemplo, o número médio de iterações efectuadas, num processo iterativo, na resolução de um conjunto de problemas teste, é uma das quantidades normalmente utilizadas para esse fim. O

conjunto inteiro dos resultados pode ser publicado num relatório e o seu acesso deve ser facultado a todos.

As conclusões a retirar dos resultados obtidos devem ser objectivas e concisas.

Outras regras muito importantes para a divulgação de resultados computacionais podem ser encontradas nos trabalhos de Jackson, Boggs, Nash e Powell (1991) e Crowder, Dembo e Mulvey (1979).

1.3 Representação de um número no computador

1.3.1 Introdução

Os números são a base dos métodos numéricos. Por esta razão vamos falar deles já no início. O primeiro tipo de erros que cometemos nos cálculos, surge do processo de representação do número. Alguns números reais são representados por uma sequência finita de dígitos. Por exemplo, a quantidade 1/8 é representada por 0.125. Normalmente podem ser representados exactamente no computador. Quando introduzimos estes valores no computador não cometemos qualquer erro. No entanto, a maior parte dos números reais são sequências infinitas de dígitos. Como exemplos muito simples, temos $1/3 = 0.333333333...$, $1/9 = 0.111111111...$, $5/7 = 0.714285714...$, $\sqrt{2} = 1.414213562...$ e $\pi = 3.141592654....$ A representação destes números no computador vai originar um erro, uma vez que só é possível introduzir um número limitado e fixo de dígitos na sua representação. Os pormenores da representação dependem do computador.

A representação habitual consiste numa sequência ordenada de dígitos. O sistema decimal, de base 10, segue este princípio. A posição de cada dígito, em relação ao ponto decimal, indica a potência de 10 correspondente.

A apresentação que se segue é válida para quantidades positivas e negativas. Assim, os números decimais com que trabalhamos podem ser colocados na forma

$$x = a_n 10^n + a_{n-1} 10^{n-1} + \cdots + a_1 10 + a_0 10^0 + a_{-1} 10^{-1} + \cdots + a_m 10^m \tag{1.1}$$

em que $n > m$ e os a_k são inteiros do conjunto $[0, 9]$; o índice k toma valores decrescentes de -1 em -1 unidade desde n até m. A notação mais conhecida é

$$x = a_n a_{n-1} \ldots a_1 a_0 \, . \, a_{-1} a_{-2} \ldots a_m. \tag{1.2}$$

Em virtude das limitações da memória dos computadores, um número x representado na forma (1.2) poderia sofrer grandes cortes. Não seria tão limitativo se esse número pudesse ser representado na forma

$$x = M \times 10^N,$$

isto é, como um número entre zero e um, M $(0 \leq M \leq 1)$, multiplicado por uma potência de 10. A quantidade M chama-se *mantissa*. Se da representação (1.1) colocarmos em evidência 10^N, com $N = n + 1$, tem-se

$$x = (a_n 10^{-1} + a_{n-1} 10^{-2} + \cdots + a_0 10^{-(n+1)} + \cdots + a_m 10^{m-N}) \times 10^N$$

ou, usando notação diferente,

$$x = 0.\, d_1 d_2 d_3 \ldots d_k d_{k+1} \ldots \times 10^N \tag{1.3}$$

em que N é um inteiro positivo ou negativo, d_1 é um inteiro do conjunto $[1, 9]$ e d_k, para $k = 2, 3, \ldots$, é inteiro do conjunto $[0, 9]$. Nesta notação a mantissa é

$$M = 0.\, d_1 d_2 d_3 \ldots d_k d_{k+1} \ldots$$

Exemplo 1.1 Dado o número $x = 80142.76013$ no sistema decimal, tem-se

$$x = 8 \times 10^4 + 0 \times 10^3 + 10^2 + 4 \times 10 + 2 \times 10^0 + 7 \times 10^{-1} + 6 \times 10^{-2} + 0 \times 10^{-3} + 10^{-4} + 3 \times 10^{-5}.$$

Como $n = 4, m = 5, a_4 = 8, a_3 = 0, a_2 = 1, a_1 = 4, a_0 = 2, a_{-1} = 7, a_{-2} = 6, a_{-3} = 0, a_{-4} = 1$ e $a_{-5} = 3$,

$$\begin{aligned} x &= (8 \times 10^{-1} + 0 \times 10^{-2} + 10^{-3} + 4 \times 10^{-4} + 2 \times 10^{-5} + 7 \times 10^{-6} + 6 \times 10^{-7} + 0 \times 10^{-8} + \\ &+ 10^{-9} + 3 \times 10^{-10}) \times 10^5 \\ &= 0.8014276013 \times 10^5 \end{aligned}$$

isto é, $d_1 = 8, d_2 = 0, d_3 = 1, d_4 = 4, d_5 = 2, d_6 = 7, d_7 = 6, d_8 = 0, d_9 = 1$ e $d_{10} = 3$. A mantissa é então igual a 0.8014276013.

Como os computadores leêm impulsos enviados por componentes eléctricos e os estados possíveis são '*on*' e '*off*' é conveniente representar os números nos computadores no *sistema binário*. A base do sistema binário é 2 e os inteiros a_k da representação são agora o 0 e o 1. Assim, o número x, no sistema binário tem a seguinte forma:

$$x = a_n 2^n + a_{n-1} 2^{n-1} + \cdots + a_1 2 + a_0 2^0 + a_{-1} 2^{-1} + \cdots + a_m 2^m$$

em que os coeficientes a_k são 0 ou 1 e $n > m$, ou ainda

$$x = 0.\, d_1 d_2 d_3 \ldots d_k d_{k-1} \ldots \times 2^N \tag{1.4}$$

em que N é um inteiro positivo ou negativo, $d_1 = 1$ e d_k, para $k = 2, 3, \ldots$ é um inteiro do conjunto $[0, 1]$.

Exemplo 1.2 O número $x = 11001.11$ do sistema binário pode ser representado na forma

$$x = 2^4 + 2^3 + 0 \times 2^2 + 0 \times 2 + 2^0 + 2^{-1} + 2^{-2}$$

ou

$$x = (0.1100111) \times 2^5$$

que corresponde a

$$\begin{aligned} x &= 2^4 + 2^3 + 1 + \frac{1}{2} + \frac{1}{2^2} \\ &= 16 + 8 + 1 + 0.5 + 0.25 = 25.75 \text{ no sistema decimal.} \end{aligned}$$

1.3.2 Formato de vírgula flutuante (normalizado)

Quando um número representado na forma (1.4) é armazenado no computador, apenas um número finito de dígitos pode ser usado para representar a mantissa. Isto é, a sequência de dígitos que representa a mantissa $M = 0.d_1d_2 \ldots d_kd_{k+1} \ldots$ deve ser finita dando origem à representação no formato de vírgula flutuante do número. Assim, define-se *formato de vírgula flutuante* de um número x à representação

$$fl(x) = \pm 0.d_1d_2 \ldots d_k \times b^e$$

em que $0.d_1d_2 \ldots d_k$ define a *mantissa* do número, com k dígitos, b é a *base* em que o número está representado, e e é o *expoente*, número inteiro limitado por dois valores, $e_{inf} < e < e_{sup}$. Na maior parte dos computadores $e_{inf} = -e_{sup}$. Os valores de k, b, e_{inf} e e_{sup} variam de computador para computador. O número k de dígitos na representação $fl(x)$ representa o *comprimento da palavra* do computador. Algumas máquinas trabalham com palavras de dois comprimentos diferentes: a mais pequena, de comprimento k, conhecida por *precisão simples* e a de comprimento duplo da primeira, que só é implementada se o utilizador o especificar, e que se chama *precisão dupla*. Os cálculos feitos com aritmética de precisão dupla ocupam duas vezes mais memória e levam pelo menos o dobro do tempo quando comparados com os cálculos em precisão simples.

O formato de vírgula flutuante obtém-se por dois processos. Um deles trunca todos os dígitos que aparecem na mantissa, à direita do d_k, da posição k, originando

$$fl_t(x) = \pm 0.d_1d_2 \ldots d_k \times 10^N$$

em que k depende do computador utilizado. Este processo é conhecido por *truncatura*. No processo de truncatura a representação do número x, no formato de vírgula flutuante *normalizado*, em que o primeiro elemento da mantissa, d_1, é diferente de zero, é obtida como sendo a quantidade mais próxima de x, que se encontra entre x e 0. O outro processo conhecido por *arredondamento*, adiciona 5 unidades ao dígito d_{k+1}, aplicando em seguida o processo de truncatura a x. Isto é equivalente a truncar, depois do dígito d_k, se o d_{k+1} é menor do que 5 (arredondamento por defeito) ou adicionar uma unidade a d_k, originando δ_k, se d_{k+1} for maior ou igual a 5 (arredondamento por excesso), truncando em seguida. Neste processo de arredondamento o $fl_a(x)$ é o formato de vírgula flutuante normalizado que está mais próximo de x. Quando $d_k = 9$, o arredondamento por excesso origina $\delta_k = 0$ e adiciona-se um a d_{k-1} originando o δ_{k-1}. Esta situação pode ocorrer noutros dígitos que apareçam a seguir na representação e o procedimento deve ser o mesmo.

Exemplo 1.3 Considere $x = 2/3$, $y = \pi$ e $z = -\sqrt{200}$. Então,
se $k = 5$

$$fl_t(x) = 0.66666 \times 10^0 \quad \text{e} \quad fl_a(x) = 0.66667 \times 10^0$$

$$fl_t(y) = 0.31415 \times 10^1 \quad \text{e} \quad fl_a(y) = 0.31416 \times 10^1,$$

se $k = 9$

$$fl_t(y) = 0.314159265 \times 10^1 \quad \text{e} \quad fl_a(y) = 0.314159265 \times 10^1$$

e se $k = 7$

$$fl_t(z) = -0.1414213 \times 10^2 \quad \text{e} \quad fl_a(z) = -0.1414214 \times 10^2.$$

1.3.3 Erro de arredondamento

O processo de substituição de um número x pelo formato de vírgula flutuante $fl(x)$ chama-se *arredondamento*. À diferença $fl(x) - x$ chama-se *erro de arredondamento*.

O número excessivamente elevado de operações aritméticas que são executadas na resolução de um problema, embora com a duração de poucos segundos, é a razão que origina a importância dos erros de arredondamento. A eficácia de qualquer algoritmo numérico deve ser verificada através do exame dos efeitos dos erros de arredondamento. Devido aos erros de arredondamento, o resultado final de um conjunto de operações é normalmente diferente do resultado exacto.

A diferença entre os resultados numérico e exacto é o erro de arredondamento *acumulado*. Os erros de arredondamento cometidos em cada etapa do cálculo são os *erros locais*, para se distinguirem do acumulado. Cada erro local propaga-se através dos restantes cálculos, vai-se acumulando aos outros erros, e a sua influência no resultado pode ser muito ampliada, o que caracteriza um algoritmo instável, ou reduzida, caracterizando um algoritmo estável.

1.3.4 Erros nos cálculos

Além dos erros de arredondamento que podemos cometer quando introduzimos dados no computador, ou quando executamos operações aritméticas, devido ao tamanho limitado da palavra do computador, outros erros podem surgir durante os cálculos numéricos. Os

- *erros crassos* ou disparates ocorrem muitas vezes e podem surgir em qualquer etapa do processo numérico. Pode ser um erro que se comete na construção do algoritmo ou na programação, no entanto, estes erros podem ser facilmente corrigidos.

- Os *erros de formulação* ou de modelação estão relacionados com uma certa tendência de alguns utilizadores não completarem, com algum rigor, o modelo matemático. Nesta situação, deve-se ter consciência do facto de que se está a trabalhar com um modelo mal construído e não adequado à realidade física. Nenhum método numérico poderá originar resultados precisos.

- Os erros nas incertezas dos dados também são conhecidos por *erros inerentes* e surgem devido ao facto de não ser possível atribuir, com exactidão, valores numéricos aos dados quando estes são obtidos por leitura experimental. Não é possível evitá-los.

- Finalmente, os *erros de truncatura* surgem do facto de ser preciso aproximar um problema de natureza contínua, como por exemplo o cálculo de $\int_a^b f(x)dx$ ou da derivada $f'(x)$, por outro de natureza discreta, por forma a ser possível determinar uma solução

que será necessariamente numérica. Outro exemplo que ilustra a ocorrência do erro de truncatura é o seguinte:

Exemplo 1.4 Pretende-se calcular e^x usando a expansão em série de Taylor da função

$$e^x = 1 + x + \frac{1}{2!}x^2 + \frac{1}{3!}x^3 + \cdots + \frac{1}{n!}x^n + \cdots$$

Se usarmos apenas os n primeiros termos da série obtém-se uma aproximação a e^x,

$$e^x \approx y(x) = 1 + x + \frac{1}{2!}x^2 + \frac{1}{3!}x^3 + \cdots + + \frac{1}{(n-1)!}x^{n-1}$$

e à diferença entre e^x e $y(x)$ chama-se o erro de truncatura.

Este tipo de erro também é inevitável, pois para calcular todos os termos da série infinita teríamos de utilizar uma quantidade infinita de tempo, que não seria viável. Voltaremos a falar do erro de truncatura mais adiante, em 1.6.

1.4 Erros absoluto e relativo

1.4.1 Definições

Os dados que intervêm nos cálculos não são, na maior parte dos casos, valores exactos. Em certos problemas, estes dados são resultados de observações experimentais, realizadas com a ajuda de instrumentos, que por muito sofisticados que sejam, não conseguem medir grandezas físicas com exactidão. Um problema deste tipo é designado por *problema físico*, e os erros que afectam estes dados, dos quais não são conhecidos os valores exactos, chamam-se erros inerentes.

Noutro tipo de problemas, os valores exactos dos dados são conhecidos mas não podem ser exactamente representados no computador, devido ao comprimento limitado da palavra do computador. Estes dados que intervêm nos cálculos vêm afectados de erros de arredondamento. A este tipo de problemas é costume designar por *problema matemático*.

Em qualquer das situações descritas, trabalha-se com valores aproximados dos dados do problema. Surge, então, a necessidade de se distinguir a categoria do valor aproximado e de se fazer o estudo da propagação desses erros, ao longo do processo numérico, e que irão influenciar o resultado final.

Seja $\bar{x}$ o valor exacto e x o seu valor aproximado que será usado nos cálculos. À diferença

$$\bar{x} - x$$

chama-se *erro absoluto* do valor x. É uma quantidade que pode ser positiva se $x < \bar{x}$ e diz-se que x aproxima por defeito, pode ser negativa se $x > \bar{x}$ e x aproxima por excesso, ou pode ser ainda nula se $x = \bar{x}$. Como para a maior parte dos problemas $\bar{x}$ é desconhecido, também não é possível calcular o erro absoluto.

Geralmente conhece-se a quantidade não negativa δ_x, tal que

$$|\bar{x} - x| \leq \delta_x, \tag{1.5}$$

a que se chama *limite superior do erro absoluto*. Desta relação pode-se concluir que o valor exacto pertence ao intervalo

$$x - \delta_x \leq \bar{x} \leq x + \delta_x.$$

À quantidade não negativa, r_x, definida por

$$r_x = \frac{|\bar{x} - x|}{|\bar{x}|}$$

chama-se *erro relativo* de x. Se δ_x for pequeno quando comparado com $\bar{x}$, então

$$r_x \approx \frac{|\bar{x} - x|}{|x|} \leq \frac{\delta_x}{|x|}. \tag{1.6}$$

O erro relativo não tem dimensão e, em geral, só é conhecido o limite superior do seu valor. A quantidade $100r_x\%$ chama-se *percentagem do erro*.

1.4.2 Fórmula fundamental dos erros

A fórmula fundamental do cálculo dos erros pode ser usada para calcular limites superiores dos erros absoluto e relativo das operações fundamentais. Assim, considere uma função de n variáveis $f(x_1, x_2, \ldots, x_n)$ e que os erros absolutos de cada um dos valores das variáveis são limitados por δ_{x_i}, para $i = 1, \ldots, n$. Usando a expansão em série de Taylor para obter uma aproximação a $f(x_1, \ldots, x_n)$, de primeira ordem,

$$f(x_1, \ldots, x_n) \approx f(\bar{x}_1, \ldots, \bar{x}_n) + \frac{\partial f}{\partial x_1}]_* \ (x_1 - \bar{x}_1) + \frac{\partial f}{\partial x_2}]_* \ (x_2 - \bar{x}_2) + \cdots + \frac{\partial f}{\partial x_n}]_* \ (x_n - \bar{x}_n)$$

em que os termos em função das segundas e restantes derivadas foram ignorados, e $\frac{\partial f}{\partial x_i}]_*$ significa que a derivada parcial deve ser calculada no ponto $(\bar{x}_1, \ldots, \bar{x}_n)$, cujos valores não são conhecidos.

Um limite superior do erro no cálculo de f é dado por

$$\delta_f = |\frac{\partial f}{\partial x_1}]_*| \ \delta_{x_1} + |\frac{\partial f}{\partial x_2}]_*| \ \delta_{x_2} + \cdots + |\frac{\partial f}{\partial x_n}]_*| \ \delta_{x_n}$$

em que

$$|x_i - \bar{x}_i| \leq \delta_{x_i}, \ \ i = 1, \ldots, n.$$

Se M_{x_i} é o majorante da derivada parcial de f em ordem a x_i, no intervalo definido por todos os valores possíveis das variáveis $\bar{x}_i$, $i = 1, \ldots, n$, isto é, para $\bar{x}_i \in [x_i - \delta_{x_i}, x_i + \delta_{x_i}]$,

$$|\frac{\partial f}{\partial x_i}]_*| \ \leq M_{x_i},$$

então

$$\delta_f \leq M_{x_1}\delta_{x_1} + M_{x_2}\delta_{x_2} + \cdots + M_{x_n}\delta_{x_n}. \tag{1.7}$$

Exemplo 1.5 Para calcular expressões para os limites superiores dos erros absoluto e relativo cometidos em $f(x_1, x_2, x_3) = x_1 x_2 x_3$ e como $\frac{\partial f}{\partial x_1} = x_2 x_3$, $\frac{\partial f}{\partial x_2} = x_1 x_3$ e $\frac{\partial f}{\partial x_3} = x_1 x_2$, tem-se

$$\delta_f = |x_2 x_3]_*| \, \delta_{x_1} + |x_1 x_3]_*| \, \delta_{x_2} + |x_1 x_2]_*| \, \delta_{x_3}.$$

Para o limite superior do erro relativo

$$\frac{\delta_f}{|x_1 x_2 x_3|} = |\frac{x_2 x_3}{x_1 x_2 x_3}| \delta_{x_1} + |\frac{x_1 x_3}{x_1 x_2 x_3}| \delta_{x_2} + |\frac{x_1 x_2}{x_1 x_2 x_3}| \delta_{x_3}$$

ou

$$r_f \leq \frac{\delta_{x_1}}{|x_1|} + \frac{\delta_{x_2}}{|x_2|} + \frac{\delta_{x_3}}{|x_3|} = r_{x_1} + r_{x_2} + r_{x_3},$$

e o erro relativo do produto de certas quantidades é inferior à soma dos erros relativos das parcelas.

Exemplo 1.6 Para $f(x_1, x_2) = x_1 - x_2$, calculem-se os limites superiores dos erros absoluto e relativo em f.

Como $\frac{\partial f}{\partial x_1} \leq M_{x_1} = 1$ e $|\frac{\partial f}{\partial x_2}| \leq M_{x_2} = 1$, tem-se

$$\delta_f = \delta_{x_1} + \delta_{x_2}$$

e para o erro relativo

$$\frac{\delta_f}{|x_1 - x_2|} = \frac{1}{|x_1 - x_2|} \delta_{x_1} + \frac{1}{|x_1 - x_2|} \delta_{x_2}$$

ou

$$r_f \leq \frac{\delta_{x_1} + \delta_{x_2}}{|x_1 - x_2|}.$$

Neste caso, quanto menor for a diferença entre x_1 e x_2, isto é, quanto mais próximas estiverem as duas quantidades, maior é o erro relativo da operação e menor será a precisão do resultado. Voltaremos a falar desta ocorrência, em 1.5.1, à qual chamaremos cancelamento subtractivo.

1.4.3 Algarismos significativos

Quando se usam os números em cálculo numérico, deve-se ter confiança nos seus valores. Por exemplo, se um dos dados num processo numérico é o comprimento x do segmento representado na figura 1.3 pode-se afirmar que o segmento tem um comprimento maior do que $10.5cm$ e menor do que $10.6cm$. Porque o extremo do lado direito ultrapassa a primeira destas marcas, a maior parte das pessoas diz que o comprimento do segmento é de $10.5cm$.

Se quiséssemos adicionar mais uma casa decimal (casa das centésimas) ao valor do comprimento, poder-se-ia avançar com o valor $10.57cm$. Mas, porque não $10.58cm$, ou mesmo $10.59cm$. Em virtude das limitações da régua, somente os três algarismos 1, 0 e 5 podem ser usados com confiança. Em relação ao quarto algarismo surgem dúvidas sobre o seu valor. A estes três primeiros algarismos chamamos *algarismos significativos* da quantidade x. Este conceito de algarismos significativos foi introduzido para marcar num número, os algarismos em que temos confiança.

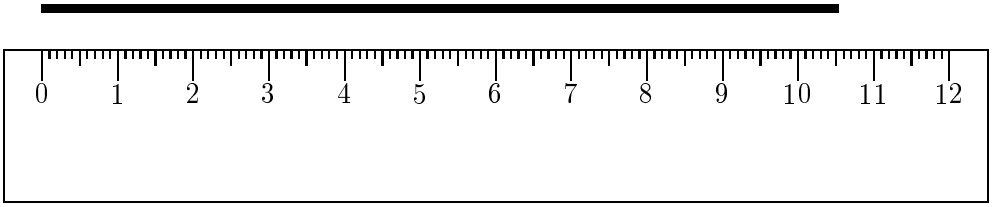


Figura 1.3: Medição do segmento

Assim, seja δ_x igual a 0.5 unidades da última casa decimal de confiança na representação de x e cuja notação simplificada é $\delta_x = (0.5)$, então são *algarismos significativos* do número x, os algarismos que se encontrem nas partes inteira e decimal de confiança de x, diferentes de zero, e os zeros que não se encontrem no número para indicar a posição da vírgula.

Exemplo 1.7 Dada a quantidade $x = 0.503142 \times 10^{-2}$ com erro absoluto inferior a 0.5×10^{-7}, quantos algarismos significativos tem x?

Como $x = 0.00503142$ e $\delta_x = 0.00000005$, a última casa decimal de x sobre a qual recai as 0.5 unidades de δ_x é o 4. Isto quer dizer que só contam como algarismos significativos os que estão à sua esquerda, e ele próprio, desde que sejam diferentes de zero e os iguais a zero que não estejam a indicar a posição da vírgula. Assim, são algarismos significativos os (contando da esquerda para a direita) 5, 0, 3, 1 e 4. Ao todo são 5.

Exemplo 1.8 Quantos algarismos significativos tem $x = 0.00050$, sabendo que $\delta_x = 0.000005$?

As 0.5 unidades de δ_x recaem sobre o último zero da direita na representação de x. Assim, contam como algarismos significativos o (a contar da esquerda para a direita) 5 e o 0. São dois.

Exemplo 1.9 Os números 0.0142, 0.000142 e 0.00000142 têm todos eles três algarismos significativos se os limites superiores dos erros absolutos forem respectivamente 0.00005, 0.0000005 e 0.000000005.

No entanto, se $\delta_x = 0.000005$ para todos eles, então o primeiro tem 3 algarismos significativos, o segundo tem dois e o último não tem nenhum.

O teorema seguinte apresenta uma relação entre o número de algarismos significativos de um valor aproximado x e o seu erro relativo.

Teorema 1.1 : Se o primeiro algarismo significativo do valor aproximado de um número é $d_1 \neq 0$ e se esse valor tem n algarismos significativos, então o seu erro relativo não excede

$$\frac{1}{d_1 \times 10^{n-1}}.$$

O corolário é para ser usado no sentido oposto, isto é, dá a indicação do número de algarismos significativos do valor aproximado x quando é conhecido um limite superior do erro relativo.

Corolário 1.2 : Se o erro relativo do valor aproximado de um número não excede

$$\frac{1}{2 \times 10^n}$$

então esse valor tem n algarismos significativos.

1.5 Efeitos numéricos da aritmética discreta

1.5.1 Cancelamento subtractivo

Alguns dos efeitos numéricos da representação dos números em formato de vírgula flutuante e da aritmética discreta dos computadores são:

1. cancelamento subtractivo,

2. estabilidade matemática,

3. estabilidade numérica.

O *cancelamento subtractivo* surge quando se pretende calcular a subtracção de duas quantidades muito próximas uma da outra. O resultado desta operação origina uma perda significativa de algarismos significativos no resultado.

Exemplo 1.10 Calcular a diferença $\sqrt{9876} - \sqrt{9875}$.
Usando uma máquina de calcular cujo mostrador apresenta 10 algarismos, tem-se

$$\sqrt{9876} - \sqrt{9875} = 0.9937806599 \times 10^2 - 0.9937303457 \times 10^2 = 0.00503142 = 0.503142 \times 10^{-2}$$

verificando-se que os dados do problema foram fornecidos com 10 algarismos significativos, supondo-se que o limite superior dos erros absolutos em cada uma das quantidades é igual a 0.5×10^{-8}, e no entanto, o resultado aparece apenas com 6 algarismos no mostrador. O limite superior do erro absoluto da operação é 1×10^{-8} ou seja $< 0.5 \times 10^{-7}$, concluindo-se que o próprio algarismo 2 do resultado não pode ser tomado como algarismo significativo.
O erro relativo desta operação é

$$r_{x-y} \leq \frac{1 \times 10^{-8}}{0.503142 \times 10^{-2}} = 1.98... \times 10^{-6} < 0.2 \times 10^{-5}$$

e, de acordo com o Corolário 1.2 o $r_{x-y} < \frac{1}{2 \times 10^5}$ donde se conclui que o resultado tem 5 algarismos significativos.

1.5.2 Estabilidade matemática

Dois dos conceitos mais importantes em Análise Numérica são o de *estabilidade matemática* e o de *estabilidade numérica*. A primeira é uma propriedade do problema matemático que vamos resolver e a segunda diz respeito ao algoritmo.

Um problema matemático diz-se *bem condicionado* se pequenas perturbações introduzidas nos dados originarem apenas pequenas alterações na solução do problema. Neste caso também se diz que o problema é *matematicamente estável*.

Por sua vez, se os resultados se apresentarem muito sensíveis a pequenas perturbações introduzidas nos dados, o problema diz-se *mal condicionado* ou *inerentemente instável*.

Para problemas bem definidos em que os resultados dependem dos dados de uma forma contínua, é possível introduzir e definir uma medida para determinar o grau do condicionamento. Essa medida chama-se *número de condição*.

Exemplo 1.11 O sistema

$$\begin{cases} x_1 + 2x_2 = 3 \\ 0.499x_1 + 1.001x_2 = 1.5 \end{cases}$$

tem como solução o vector $x = (1.0, 1.0)^T$. O número de condição deste problema é $\|A\|_1 \|A^{-1}\|_1 = 3001$, bastante elevado, o que nos leva a concluir que o sistema é mal condicionado (veja-se em 3.2.2). De facto, se o elemento da posição $(2,1)$ for perturbado de uma quantidade igual a 0.001, que equivale a sofrer uma perturbação da ordem de 2×10^{-3}, em termos relativos, o sistema perturbado passa a ser

$$\begin{cases} x_1 + 2x_2 = 3 \\ 0.5x_1 + 1.001x_2 = 1.5 \end{cases}$$

cuja solução é agora $\bar{x} = (3.0, 0.0)^T$. O resultado sofreu perturbações, em termos relativos, da ordem de $|1.0 - 3.0|/|1.0| = 3.0$ e $|1.0 - 0.0|/|1.0| = 1.0$. Os resultados são muito sensíveis a pequenas perturbações nos dados.

No caso da resolução de sistemas de equações lineares o número de condição está relacionado com a proximidade da matriz dos coeficientes do sistema em relação a uma matriz singular.

Exemplo 1.12 A equação não linear

$$x^2 - 2.029x + 1.028 = 0$$

tem as seguintes raízes, $x_1 = 1.049288648$ e $x_2 = 0.979711352$.

No entanto, se perturbarmos o termo independente de 0.001, que corresponde a um erro relativo da ordem de 9.7×10^{-4}, a equação perturbada passa a ser,

$$x^2 - 2.029x + 1.029 = 0$$

e as suas raízes são 1.029 e 1 respectivamente. Os resultados sofreram perturbações da ordem de 0.019 e 0.021, em termos relativos. Os erros nos resultados sofreram um aumento em relação aos erros dos dados. O problema é assim mal condicionado.

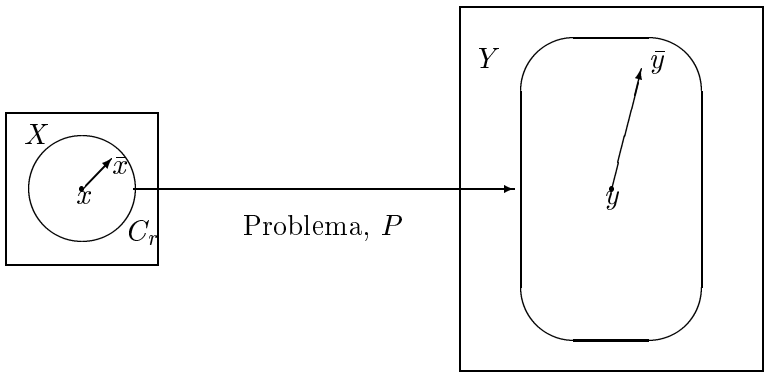


Figura 1.4: Esquema do comportamento de um problema mal condicionado

É possível ilustrar geometricamente o problema da estabilidade matemática através de um esquema muito simples. Um problema matemático P consiste numa transformação de um espaço X de dados possíveis num outro espaço Y de resultados. A figura 1.4 apresenta essa transformação. Se os dados x forem perturbados, dando origem a $\bar{x} \in C_r$, valores do círculo de centro x e raio r, um problema mal condicionado produz resultados $\bar{y}$ da figura oval, fortemente perturbados em relação a y.

1.5.3 Estabilidade numérica

A estabilidade numérica é uma propriedade dos algoritmos. Podem existir várias maneiras de organizar os cálculos que levam à resolução de um problema matemático. Os diferentes algoritmos competem uns com os outros no que diz respeito à sua complexidade (número de operações aritméticas necessárias), à quantidade de memória necessária para armazenar os dados e os resultados intermédios e aos limites dos erros cometidos e que se propagam de operação para operação, afectando os resultados finais. Processos numéricos distintos, para resolver o mesmo problema matemático, podem comportar-se de maneira diferente em relação à propagação dos erros de arredondamento. Esta sensibilidade maior ou menor relativa às operações de arredondamento chama-se *estabilidade numérica*. Esta estabilidade depende da maneira como estão organizados os cálculos.

Se um resultado intermédio está contaminado por erros de arredondamento, esses erros irão influenciar todos os resultados seguintes que dependem daquele resultado intermédio. Na prática, cada resultado intermédio introduz novos erros de arredondamento. Os erros de arredondamento *propagam-se* de operação para operação indo influenciar o resultado final. Se estes erros influenciam fortemente o resultado final o algoritmo diz-se *numericamente instável*.

Se, por efeitos estatísticos, os erros introduzidos pelos resultados intermédios, em alguns casos, vierem a cancelar parcialmente uns com os outros, o efeito resultante no resultado final torna-se insignificante. Os algoritmos que satisfazem esta propriedade são considera-

dos *numericamente estáveis*. Exemplos de algoritmos estáveis são os métodos iterativos, dos quais veremos muitos casos ao longo do livro. Um dos métodos directos instáveis mais conhecido é o método de eliminação de Gauss, para a resolução de sistemas de equações lineares.

Exemplo 1.13 A resolução do sistema linear

$$\begin{cases} 0.005x_1 + x_2 = 0.5 \\ x_1 + x_2 = 1 \end{cases}$$

pela implementação do algoritmo de eliminação de Gauss (veja-se em 3.4.2) origina o sistema equivalente

$$\begin{aligned} 0.005x_1 + x_2 &= 0.5 \\ - 199x_2 &= -99 \end{aligned}$$

e o resultado é o vector $x = (0.6, 0.497)^T$, se usarmos aritmética com três algarismos significativos nos cálculos.

No entanto a implementação do algoritmo de eliminação de Gauss com pivotagem parcial (também no Capítulo 3, em 3.4.4), origina, numa primeira fase o sistema

$$\begin{aligned} x_1 + x_2 &= 1 \\ 0.005x_1 + x_2 &= 0.5 \end{aligned}$$

e em seguida

$$\begin{aligned} x_1 + x_2 &= 1 \\ + 0.995x_2 &= 0.495, \end{aligned}$$

sendo a solução o vector $x = (0.503, 0.497)^T$ e que é a solução exacta arredondada para três algarismos significativos.

O algoritmo de eliminação de Gauss com pivotagem parcial para a resolução de sistemas de equações lineares, que veremos no Capítulo 3, é um exemplo de um método estável, uma vez que não induz instabilidade.

Podemos concluir que para a resolução de um problema bem condicionado e dependendo da maneira como se encontram organizadas as operações, um algoritmo pode ser estável, se os erros inerentes e de arredondamento não se acumularem exageradamente, ou instáveis, se esses mesmos erros se acumularem e se propagarem exageradamente por forma a afectarem significativamente os resultados finais. Por outro lado, se o problema for mal condicionado, nenhum algoritmo conseguirá atenuar a acumulação dos erros inerentes e de arredondamento.

1.5.4 A análise dos erros

Um dos critérios usados para decidir sobre a eficiência dos algoritmos numéricos está relacionado com as estimativas e limites dos erros cometidos durante todo o processo de cálculo. A análise dos erros de arredondamento pode ser feita por um processo *directo*, *inverso*, *intervalar* ou *estatístico*.

A *análise directa dos erros* consiste em estimar os limites dos erros cometidos em cada passo do algoritmo numérico seguindo a sequência de operações executadas.

A *análise inversa dos erros* toma o resultado final calculado e que vem afectado de erros, como sendo o resultado exacto de um problema perturbado que resulta do problema original perturbando os dados do problema. O objectivo é pois determinar o problema perturbado cuja solução é conhecida.

A *análise intervalar* é um processo que determina a solução do problema e um limite dos erros cometidos ao longo do processo numérico, operando com intervalos e fazendo uso da aritmética intervalar. A aritmética intervalar tem os seus próprios conceitos, definições e terminologia.

Finalmente, a *análise estatística dos dados* usa técnicas e conceitos estatísticos para estimar os valores mais prováveis para o resultado de um problema. A distribuição estatística mais usada para descrever o comportamento dos erros ao longo do processo numérico é a distribuição Normal com média zero e desvio padrão um. Em certas situações também é usada a distribuição uniforme.

1.6 Erros de truncatura

1.6.1 Introdução

Além dos erros de arredondamento que se cometem durante os cálculos, outros erros, chamados *erros de truncatura*, surgem com a utilização de certos métodos numéricos para a resolução do problema matemático. Assim, cometem-se erros de truncatura quando se usam *métodos de discretização* e *métodos iterativos*.

1.6.2 Métodos de discretização

Quando o problema matemático a resolver envolve conceitos contínuos, por exemplo, o cálculo de um integral de uma função num certo domínio e a resolução de uma equação diferencial, a sua resolução numérica transforma-o num problema discreto. A solução do problema discreto é calculada por um processo muito simples e rápido e é obtida com grande precisão. Esta solução é considerada como uma aproximação à solução do problema original. Os métodos numéricos deste tipo chamam-se *métodos de discretização* e o problema diz-se que foi discretizado.

Exemplo 1.14 O cálculo do integral $\int_{1.0}^{1.2} \sqrt{x}\,dx$ pode ser discretizado originando o seguinte problema:

- Calcular
$$\frac{0.05}{3}[f(1.0) + 4f(1.05) + 2f(1.1) + 4f(1.15) + f(1.2)].$$
Como $f(1.0) = 1, f(1.05) = 1.024695, f(1.1) = 1.0488088, f(1.15) = 1.0723805$ e $f(1.2) = 1.0954451$ a solução do problema discretizado é 0.2096894.

Assim,

$$\int_{1.0}^{1.2} \sqrt{x}dx \approx 0.2096894.$$

Como o valor exacto do integral é 0.209689425, com nove casas decimais, o erro de truncatura cometido é inferior a 0.3×10^{-7}.

Exemplo 1.15 A equação diferencial $y'(x) = -xy^2(x)$, com $y(0) = 2$, pode ser resolvida através da sua discretização. O problema discreto correspondente determina aproximações $y_1, y_2, y_3, \ldots$ à função $y(x)$ nos pontos $x_1, x_2, x_3, \ldots$, usando a equação algébrica

$$y_{i+1} = y_i + (x_{i+1} - x_i)y'(x_i), \;\; i = 0, 1, 2, \ldots$$

Assim, os correspondentes valores de y são os apresentados na $2^{a\cdot}$ coluna da tabela,

x_i	y_i	$y(x_i)$
0.0	2	2
0.1	2	1.98019802
0.2	1.96	1.923076923
0.3	1.883168	1.834862385
0.4	1.776778349	1.724137931
0.5	1.650500697	1.6

A solução exacta da equação diferencial é a função $y(x) = \frac{2}{(1+x^2)}$, que toma os valores apresentados na $3^{a\cdot}$ coluna da tabela, nos pontos $0.1, 0.2, 0.3, 0.4$ e 0.5. De acordo com os valores apresentados, o erro de truncatura deste processo de discretização, a diferença entre o valor aproximado e o valor exacto, é inferior a 0.06.

1.6.3 Métodos iterativos

Quando se usam métodos iterativos para calcular a solução de um problema matemático, essa solução é obtida, teoricamente, ao fim de um número infinito de operações. Os *métodos iterativos* são métodos definidos por uma *equação iterativa*,

$$x_{k+1} = f(x_k), \;\; k = 1, 2, \ldots$$

a partir da qual se constrói, passo a passo, uma sequência de aproximações à solução. Cada passo tem o nome de *iteração*. O índice k indica a iteração em que nos encontramos. O processo iterativo tem de ser iniciado com uma aproximação inicial à solução, x_1. O processo iterativo diz-se convergente se

$$lim_{k\to\infty} x_k = x^*$$

sendo x^* a solução do problema, isto é, o ponto de acumulação de $f(x)$: $x^* = f(x^*)$. Se, para que se verifique convergência, a aproximação inicial, x_1, deve estar próxima da solução, o método iterativo diz-se *localmente convergente*. Se, por outro lado, for possível provar convergência para qualquer valor inicial, x_1, então o método é *globalmente convergente*.

Se, teoricamente, a solução exacta seria conseguida no limite, ao fim de um número infinito de iterações, na prática, este tipo de implementação tem um grande inconveniente.

Porque os nossos recursos são limitados, os processos iterativos devem ser terminados ao fim de um número finito de operações, desde que se tenha a garantia de que a aproximação calculada se encontra muito perto da solução exacta. As condições impostas para que esta situação se verifique definem aquilo a que normalmente se chama *critério de paragem do processo iterativo*. O erro de truncatura cometido, nestes casos, dá-nos a diferença entre o valor conseguido, quando da paragem do processo iterativo, e o valor exacto que seria obtido no limite. Seguem-se dois exemplos muito simples que ilustram esta situação.

Exemplo 1.16 O cálculo do valor de e pode ser feito através do seguinte processo iterativo:

$$x_{k+1} = f(x_k) = x_k + \frac{1}{k!}, \quad k = 1, 2, \ldots$$

sendo $x_1 = 1$. Assim,

$$x_2 = x_1 + 1 = 2$$

$$x_3 = x_2 + \frac{1}{2!} = 2.5$$

$$x_4 = x_3 + \frac{1}{3!} = 2.666666667$$

$$\ldots$$

$$x_8 = 2.718253969$$

Se o processo iterativo terminar nesta iteração, a aproximação conseguida 2.718253969 tem um erro de truncatura inferior a 3×10^{-5}.

Exemplo 1.17 A solução da equação $x^2 = 2$ pode ser calculada através do processo iterativo:

$$x_{k+1} = f(x_k) = \frac{1}{2}(x_k + \frac{2}{x_k}), \quad k = 1, \ldots$$

com $x_1 = 2$. As quatro primeiras iterações dão os seguintes valores

$$x_2 = \frac{1}{2}(x_1 + \frac{2}{x_1}) = \frac{1}{2}(2 + \frac{2}{2}) = 1.5$$

$$x_3 = \frac{1}{2}(x_2 + \frac{2}{x_2}) = \frac{1}{2}(1.5 + \frac{2}{1.5}) = 1.416666667$$

$$x_4 = \frac{1}{2}(x_3 + \frac{2}{x_3}) = \frac{1}{2}(1.416666667 + \frac{2}{1.416666667}) = 1.414215686$$

$$x_5 = 1.414213562$$

sendo o x_5 uma aproximação com erro de truncatura inferior a 4×10^{-10}.

1.7 Problemas

1. O resultado de uma operação não tem necessariamente o mesmo número de algarismos significativos do que as parcelas. Comprove a afirmação, calculando

$$x + y \text{ com } x = 0.123 \times 10^4 \text{ e } y = 0.456 \times 10^{-3}.$$

2. Para $x = 0.433 \times 10^2$, $y = 0.745 \times 10^0$ e $z = 0.100 \times 10^1$, calcule usando aritmética de três algarismos significativos

   - $x + y$,
   - $\frac{y}{x}$,
   - xz.

   Quantos algarismos significativos apresentam os resultados? Estime os erros de arredondamento cometidos.

3. Calcule um limite superior do erro absoluto no cálculo da expressão

$$f(x, y, z) = \frac{2xy}{x^2 + z}$$

   sabendo que são usados os seguintes valores aproximados

$$x = 3.1415 \text{ de } \pi$$

$$y = 1.732 \text{ de } \sqrt{3}$$

$$z = 1.4142 \text{ de } \sqrt{2}.$$

   Estime também o erro relativo em f. Quantos algarismos significativos apresenta o resultado obtido?

4. Reformule a expressão

$$\frac{1 - cos(x)}{x}$$

   por forma a tornar-se estável para $x \neq 0$ e $|x| << 1$.

5. A fórmula resolvente para equações do $2^{o.}$ grau,

$$x_1, x_2 = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

   pode ser colocada na seguinte forma:

$$x_1 = \frac{-b - sgn(b)\sqrt{b^2 - 4ac}}{2a}$$

   e

$$x_2 = \frac{c}{ax_1}$$

   Discuta as duas formulações.

   Calcule, pelos dois processos, as raízes de $x^2 - 320x + 16 = 0$, usando aritmética de quatro algarismos significativos no formato de vírgula flutuante normalizado.

6. Para valores de x muito grandes, calcule

$$R = \frac{1}{\sqrt{x}} - \frac{1}{\sqrt{x+1}}.$$

Apresente outra formulação que não origine cancelamento subtractivo.

Usando aritmética de três algarismos significativos, calcule o valor de R, a partir da fórmula dada e também a partir da formulação encontrada. Considere $x = 0.100 \times 10^5$.

Capítulo 2

Solução de uma equação não linear

2.1 Introdução

2.1.1 Forma geral do problema

Uma equação não linear na variável x é representada na forma

$$f(x) = 0 \tag{2.1}$$

em que $f : \mathbf{R} \to \mathbf{R}$ é uma função não linear em x. A variável x diz-se *independente* e a variável $y = f(x)$ é a *variável dependente*. Resolver a equação (2.1) consiste em calcular as suas raízes, ou, determinar os zeros da função $f(x)$. Se representarmos graficamente a função $f(x)$ no plano XOY, os pontos de intersecção da curva $f(x)$ com o eixo dos $X's$ definem as raízes reais de (2.1). Veja-se o exemplo da figura 2.2. Pode-se esperar que uma equação do tipo (2.1) tenha raízes reais e/ou complexas.

2.1.2 Características do problema

Existem dois tipos de equações não lineares. As *algébricas* que envolvem apenas as operações aritméticas básicas, de que a forma polinomial é um caso particular,

$$p_n(x) = a_0 x^n + a_1 x^{n-1} + a_2 x^{n-2} + \cdots + a_{n-1} x + a_n = 0, \tag{2.2}$$

sendo os coeficientes a_i, $i = 0, \ldots, n$ números reais ou complexos, e as *transcendentes* que envolvem funções trigonométricas, exponenciais, logarítmicas, entre outras. São exemplos, os casos

$$f(x) = x - e^{-x} = 0,$$

$$f(x) = x + ln(x) = 0$$

e

$$f(x) = (2x + 1)^2 - 4cos(\pi x) = 0.$$

Se, para um dado valor da variável independente x, se pretende calcular o correspondente valor de $f(x)$, o problema diz-se *directo*. No entanto, se o objectivo é determinar o(s) valor(es) de x que satisfaz(em) a equação representada em (2.1), então o problema diz-se *inverso*. O problema directo não oferece qualquer dificuldade, apenas o problema inverso requer, na grande maioria dos casos, a utilização de um método numérico.

2.1.3 Métodos iterativos. Razão de convergência

A maior parte dos métodos numéricos para a resolução do problema definido em (2.1) pertence à classe dos métodos iterativos.

Define-se *sequência* de números, $\{x_k\}$, $k = 1, 2, \ldots$, como sendo uma transformação do conjunto dos inteiros positivos no conjunto dos números reais. O número real associado ao inteiro k é designado por x_k.

Uma sequência diz-se *recursiva* se o elemento de ordem k da sequência é definido por uma função de um ou de vários elementos anteriores a ele,

$$x_k = F(k, x_{k-1}, x_{k-2}, \ldots),$$

e os primeiros elementos da sequência têm de ser dados explicitamente.

Uma sequência diz-se definida por *iteração* se a função F é independente de k. A sequência resultante

$$x_k = F(x_{k-1}, \ldots)$$

chama-se *sequência iterativa* gerada por F.

A sequência iterativa diz-se *convergente*[1] se

$$lim_{k\to\infty} x_k = x^*$$

sendo x^* um ponto fixo da equação, isto é, $x^* = F(x^*)$.

Um método iterativo é definido por uma *equação iterativa*, com a qual construímos aproximações à solução do problema. A implementação da equação iterativa obriga ao conhecimento de uma aproximação inicial e à definição de um conjunto de condições que nos dêem a garantia de que a aproximação calculada, numa certa iteração, se encontra suficientemente perto da solução. Quando estas condições forem verificadas, podemos parar o processo iterativo.

Existem questões muito importantes relacionadas com um método iterativo, para as quais convém obter resposta antes de se iniciar o processo:

[1]Definição : A sequência de números x_k converge para o limite x^* se, dada uma tolerância $\epsilon > 0$, existe um índice $N = N(\epsilon)$ de tal modo que para todos os $k \geq N$ temos $|x_k - x^*| \leq \epsilon$.

Para simplificar escrevemos:

$$\lim_{k\to\infty} x_k = x^*.$$

1. A primeira diz respeito à convergência - interessa saber se o método iterativo converge ou não para a solução procurada. Devem ser analisadas as condições necessárias e/ou suficientes de convergência do método.

2. Tendo a garantia de que o método vai convergir para a solução, a segunda questão a colocar consiste em saber qual a *razão de convergência*:

   - a convergência da sequência $\{x_k\}$ diz-se *linear* se

   $$lim_{k\to\infty}\frac{|x^* - x_{k+1}|}{|x^* - x_k|} = L < 1, \tag{2.3}$$

   sendo L uma constante positiva;

   - a convergência diz-se *superlinear* se

   $$lim_{k\to\infty}\frac{|x^* - x_{k+1}|}{|x^* - x_k|^p} = L, \text{ com } p = 1.618 \text{ ou } lim_{k\to\infty}\frac{|x^* - x_{k+1}|}{|x^* - x_k|} = 0; \tag{2.4}$$

   - a convergência diz-se *quadrática* se

   $$lim_{k\to\infty}\frac{|x^* - x_{k+1}|}{|x^* - x_k|^2} = L. \tag{2.5}$$

3. A implementação de um método iterativo exige a realização de um número infinito de operações para se chegar à solução. No entanto, face aos recursos limitados disponíveis, o processo iterativo tem de ser terminado após um número finito de operações. Esta paragem deve ser feita com a ajuda de condições, que, sendo verificadas, nos dão a garantia de que estamos perto da solução. O valor obtido na última iteração é a melhor aproximação calculada. Estas condições definem o que daqui para a frente será designado por *critério de paragem* de um processo iterativo. Falaremos do critério de paragem, para o problema $f(x) = 0$, em 2.3.

Os métodos para resolver o problema (2.1) podem ser classificados em dois grandes grupos. O grupo dos *métodos de encaixe* é caracterizado por definir, em cada iteração, um intervalo que contém a raiz e construir, para a iteração seguinte, outro intervalo encaixado neste e que continue a conter a raiz. Os intervalos, como aparecem encaixados uns nos outros, têm amplitudes sucessivamente menores. Como exemplos deste tipo de métodos temos o *método da bissecção* e o da *falsa posição*.

O segundo grupo engloba métodos como o de *Newton-Raphson*, o da *secante*, o de *Laguerre* e o do *ponto fixo*. São *métodos de intervalo aberto* onde não é necessário definir um intervalo que contenha a raiz. O processo iterativo pode ser iniciado com uma única aproximação à raiz, ou mesmo duas. A convergência dos métodos deste grupo depende dos valores iniciais atribuídos na primeira iteração.

2.1.4 Índice de algoritmos

Neste Capítulo são apresentados 11 algoritmos. A listagem é a seguinte:

Algoritmo 2.1 Regra de Ruffini-Horner para o cálculo do valor de um polinómio

Algoritmo 2.2 Regra de Ruffini-Horner para o cálculo dos valores de um polinómio e da sua primeira derivada

Algoritmo 2.3 Regra de Ruffini-Horner para o cálculo dos valores de um polinómio e das suas primeira e segunda derivadas

Algoritmo 2.4 Critério de paragem do processo iterativo

Algoritmo 2.5 Cálculo de uma aproximação à precisão do computador

Algoritmo 2.6 Método da bissecção para o cálculo de uma raiz real

Algoritmo 2.7 Método da falsa posição para o cálculo de uma raiz real

Algoritmo 2.8 Método da secante para o cálculo de uma raiz real

Algoritmo 2.9 Método de Newton-Raphson para o cálculo de uma raiz real

Algoritmo 2.10 Método de Laguerre para o cálculo de uma raiz complexa (ou real)

Algoritmo 2.11 Método do ponto fixo para o cálculo de uma raiz real (inclui aceleração de Aitken)

Os três primeiros contêm a regra de Ruffini-Horner para calcular respectivamente o valor de um polinómio, o valor do polinómio e da primeira derivada, e do polinómio e das suas primeira e segunda derivadas. O algoritmo 2.4 apresenta um critério para a paragem de um método iterativo e o algoritmo 2.5 calcula o menor número positivo múltiplo de 2 no formato de vírgula flutuante. Os seis restantes correspondem a métodos iterativos para calcular raízes de equações não lineares. Os algoritmos 2.6 e 2.7 são de encaixe e os restantes de intervalo aberto. Os métodos da secante e de Newton-Raphson, respectivamente, algoritmos 2.8 e 2.9 podem ser usados para calcular raízes complexas, desde que seja introduzida a aritmética complexa na sua implementação. O algoritmo 2.10 calcula raízes complexas de polinómios.

2.2 Equações algébricas

2.2.1 Regra de Ruffini-Horner

Uma equação algébrica é uma equação que envolve polinómios da forma

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

O teorema fundamental da Álgebra[2] diz que um polinómio de grau n tem exactamente n zeros e se os coeficientes forem reais podem existir pares de zeros conjugados.

Em certos métodos há necessidade de calcular o valor do polinómio, $p_n(x)$, para $x = x_i$. Não é eficiente calcular um termo de cada vez e adicioná-los no fim. Aliás, este processo é o mais utilizado, exigindo um total de $2n-1$ multiplicações e n adições, sendo n o grau do polinómio.

Se o polinómio for escrito na forma

$$p_n(x) = (\ldots(((a_0x + a_1)x + a_2)x + a_3)\ldots a_{n-1})x + a_n$$

podemos obter o valor de $p_n(x)$ para $x = x_i$, $p_n(x_i)$, recursivamente, utilizando a *regra de Ruffini-Horner*[3] que necessita, apenas, de n multiplicações e n adições.

O processo recursivo consiste em calcular uma sequência recursiva $\{p_k\}$, a partir da fórmula,

$$p_k = p_{k-1}x_i + a_k \text{ para } k = 1, 2, \ldots, n \tag{2.6}$$

em que $p_0 = a_0$ e os a_k, $k = 0, 1, \ldots, n$ são os coeficientes do polinómio. O último valor da sequência, p_n, é precisamente o valor de $p_n(x_i)$. Quando comparado com a técnica do cálculo termo a termo, este processo recursivo é tanto mais rápido quanto maior for n. O algoritmo 2.1 representa este processo recursivo.

Algoritmo 2.1 :

1. `ler` n, x_i, `para` $k = 0, 1, \ldots, n$ `ler` a_k `e fazer` $p_0 = a_0$
2. `para` $k = 1, 2, \ldots, n$ `fazer` $p_k = p_{k-1}x_i + a_k$
3. `terminar com` $p_n(x_i) \leftarrow p_n$.

Veremos como é possível construir uma relação de recorrência para o cálculo da primeira derivada do polinómio, $p'_n(x)$, para $x = x_i$, utilizando novamente a regra de Horner. As quantidades p_k, $k = 0, 1, 2, \ldots, n-1$, calculadas pela fórmula (2.6) são os coeficientes do polinómio quociente, $q_{n-1}(x)$, que se obtém quando se divide $p_n(x)$ por $x - x_i$. Assim,

$$p_n(x) = (x - x_i)q_{n-1}(x) + \text{resto} \tag{2.7}$$

em que $q_{n-1}(x) = p_0x^{n-1} + p_1x^{n-2} + \cdots + p_{n-1}$. Derivando uma vez a equação (2.7), em ordem a x, obtém-se

$$p'_n(x) = (x - x_i)q'_{n-1}(x) + q_{n-1}(x)$$

[2]Teorema fundamental da Álgebra: Seja $p_n(x)$ um polinómio de grau $n \geq 1$, isto é, $p_n(x) = a_0x^n + a_1x^{n-1} + \cdots + a_{n-1}x + a_n$, em que os a_i são números reais ou complexos e $a_n \neq 0$. Então existem n constantes únicas ξ_i, $i = 1, \ldots, n$ tais que $p_n(\xi_i) = 0$ e

$$p_n(x) = a_0(x - \xi_1)(x - \xi_2)\ldots(x - \xi_n).$$

Os ξ_i podem não ser todos distintos ou todos reais.

[3]P. Ruffini viveu entre 1765 e 1822 e publicou dois trabalhos relacionados com a regra de Ruffini para o cálculo de raízes, respectivamente em 1804 e 1813. Por esta sua descoberta Ruffini recebeu uma medalha de ouro. Horner publicou um trabalho em 1819 sobre a mesma regra e pensa-se que a tenha desenvolvido em total desconhecimento do trabalho de Ruffini (Goldstine (1977)).

e substituindo x por x_i, vem

$$p'_n(x_i) = q_{n-1}(x_i).$$

A sequência recursiva $\{q_k\}$, pode ser calculada a partir da fórmula,

$$q_k = q_{k-1}x_i + p_k \text{ com } p_k = p_{k-1}x_i + a_k \text{ para } k = 1, 2, \ldots, n-1 \tag{2.8}$$

em que $p_0 = q_0 = a_0$.

O último valor da sequência, q_{n-1}, dá o valor de $p'_n(x_i)$. Os valores de $p_n(x_i)$ e $p'_n(x_i)$ podem ser obtidos simultaneamente, de acordo com o algoritmo 2.2.

Algoritmo 2.2 :

1. `ler` n, x_i `e para` $k = 0, 1, \ldots, n$ `ler` a_k `e fazer` $p_0 = a_0$ `e` $q_0 = a_0$
2. `para` $k = 1, 2, \ldots, n-1$ `fazer` $p_k = p_{k-1}x_i + a_k$ `e` $q_k = q_{k-1}x_i + p_k$
3. `fazer` $p_n = p_{n-1}x_i + a_n$
4. `terminar com` $p_n(x_i) \leftarrow p_n$ `e` $p'_n(x_i) = q_{n-1}(x_i) \leftarrow q_{n-1}$.

Derivando duas vezes a equação (2.7) tem-se

$$p''_n(x) = (x - x_i)q''_{n-1}(x) + 2q'_{n-1}(x)$$

que, para $x = x_i$, dá

$$p''_n(x_i) = 2q'_{n-1}(x_i).$$

Como o cálculo de $q'_{n-1}(x_i)$ pode ser feito, novamente, pela regra de Horner, temos então um processo recursivo para o cálculo de $q'_{n-1}(x_i)$. Assim, para $k = 1, 2 \ldots, n-2$

$$r_k = r_{k-1}x_i + q_k \text{ com } q_k = q_{k-1}x_i + p_k \text{ e } p_k = p_{k-1}x_i + a_k \tag{2.9}$$

em que $p_0 = q_0 = r_0 = a_0$. O último valor da sequência, r_{n-2}, é metade de $p''_n(x_i)$. O algoritmo para o cálculo simultâneo de $p_n(x_i)$, $p'_n(x_i)$ e $p''_n(x_i)$ é o seguinte:

Algoritmo 2.3 :

1. `ler` n, x_i `e para` $k = 0, 1, \ldots, n$ `ler` a_k `e fazer` $p_0 = a_0$, $q_0 = a_0$ `e` $r_0 = a_0$
2. `para` $k = 1, 2, \ldots, n-2$ `fazer` $p_k = p_{k-1}x_i + a_k$, $q_k = q_{k-1}x_i + p_k$ `e` $r_k = r_{k-1}x_i + q_k$
3. `fazer` $p_{n-1} = p_{n-2}x_i + a_{n-1}$ `e` $q_{n-1} = q_{n-2}x_i + p_{n-1}$
4. `fazer` $p_n = p_{n-1}x_i + a_n$
5. `terminar com` $p_n(x_i) \leftarrow p_n$, $p'_n(x_i) \leftarrow q_{n-1}$ `e` $p''_n(x_i) \leftarrow 2r_{n-2}$.

2.2.2 Condicionamento das raízes

Em relação ao problema inverso representado em (2.2), determinação dos zeros de um polinómio, vamos analisar a sua estabilidade matemática. Um polinómio pode ter uns zeros bem condicionados e outros mal condicionados. Assim, vamos ver os efeitos resultantes nos zeros do polinómio de ligeiras perturbações introduzidas nos coeficientes.

Um zero de um polinómio, x^*, diz-se *isolado* se existe uma única raiz de $p_n(x) = 0$ igual a x^*, isto é, se for verificado o seguinte:

$$p_n(x) = (x - x^*)q_{n-1}(x)$$

$$p_n'(x^*) \neq 0, \;\; p_n''(x^*) \neq 0, \;\; \ldots$$

Um zero x^* de $p_n(x)$ diz-se que tem *multiplicidade* m se existirem m raízes de $p_n(x) = 0$ iguais a x^*. Neste caso, verificam-se as seguintes condições:

$$p_n(x) = (x - x^*)^m q_{n-m}(x)$$

$$p_n'(x^*) = 0, \;\; p_n''(x^*) = 0, \ldots, \;\; p_n^{(m-1)}(x^*) = 0, \;\; p_n^{(m)}(x^*) \neq 0, \ldots$$

Perturbando cada um dos coeficientes do polinómio, a_i, $i = 0, 1, \ldots, n$, obtém-se o polinómio perturbado

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

em que

$$|a_i - a_i^\epsilon| \leq \epsilon, \;\; i = 0, 1, \ldots, n$$

sendo ϵ uma quantidade pequena e positiva. Ao zero isolado x^* de $p_n(x)$ corresponde o zero $x^*(\epsilon)$ do polinómio perturbado, $p_n^\epsilon(x)$. A variação induzida em x^* é dada por

$$x^* - x^*(\epsilon) = -\frac{K\epsilon}{p_n'(x^*)} + \mathcal{O}(\epsilon^2) \tag{2.10}$$

em que K é uma constante e

$$lim_{\epsilon \to 0} \mathcal{O}(\epsilon^2) = 0.$$

Assim, quando $p_n'(x^*)$ toma valores próximos de zero, a variação em x^* é enorme e o zero diz-se *mal condicionado*.

Se a análise da perturbação do zero x^*, diz respeito a um zero de multiplicidade m, então essa variação é agora dada por

$$x^* - x^*(\epsilon) = \{-\frac{m!\, K\epsilon}{p_n^{(m)}(x^*)}\}^{\frac{1}{m}} + \mathcal{O}(\epsilon^2). \tag{2.11}$$

Neste caso, o zero é *mal condicionado* ou porque a derivada de ordem m, $p_n^{(m)}(x^*)$, tem um valor próximo de zero, ou porque a sua multiplicidade m é elevada. Repare-se no termo $m!$ no numerador e na raiz índice m da expressão: quanto maior for m maior poderá ser o valor da expressão.

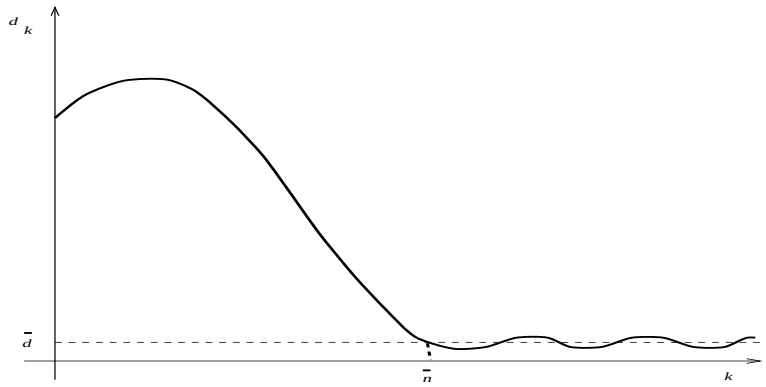


Figura 2.1: Comportamento de d_k ao longo das iterações

2.3 Critério de paragem

Vimos já que uma das questões levantadas pela implementação de um algoritmo do tipo iterativo está relacionada com a paragem deste processo. Esta deve-se dar quando a aproximação calculada, na presente iteração, estiver já muito próxima da solução, de tal forma que o erro cometido com a utilização dessa aproximação é inferior a uma tolerância especificada. São duas as condições que se devem incluir no critério de paragem e que, sendo verificadas, dão garantia da situação descrita.

- A condição que dá uma estimativa do erro relativo da aproximação calculada é

$$d_k = \frac{|x_{k+1} - x_k|}{|x_{k+1}|} \leq \varepsilon_1 \tag{2.12}$$

  em que x_{k+1} é a última aproximação calculada, x_k a aproximação da iteração anterior e ε_1 uma quantidade pequena e positiva, que estima a precisão dessa aproximação.

  A quantidade d_k da condição (2.12) tem um comportamento semelhante ao indicado na figura 2.1. Nas primeiras iterações pode, eventualmente, aumentar um pouco, mas começa logo a descer, à medida que se vão fazendo iterações, caso o processo esteja a convergir. Na prática, o seu valor não desce abaixo de $\bar{d}$ (ver na figura 2.1), uma vez que este valor corresponde à acumulação dos erros de arredondamento que se verificou ao longo do processo numérico. Atingindo-se a iteração $\bar{n}$, a partir da qual não é possível fazer baixar o valor de d_k, o processo iterativo pode ser terminado, uma vez que já se atingiu o limite da precisão e não é possível melhorar a aproximação calculada.

  Se o processo iterativo estiver a convergir para $x = 0$, ou passar por valores muito próximos de zero, a quantidade d_k em (2.12) deve ser substituída por

$$|x_{k+1} - x_k|.$$

- A segunda condição, que deve fazer parte do critério de paragem do algoritmo, resulta do facto de se procurar uma raiz de $f(x) = 0$. Assim, se a condição

$$|f(x_{k+1})| \leq \varepsilon_2 \tag{2.13}$$

for verificada, com ε_2 pequeno e positivo, tem-se a garantia da proximidade de uma raiz.

As duas condições (2.12) e (2.13) devem ser verificadas simultaneamente para ser possível garantir a convergência para uma raiz.

Se o processo for muito lento ou estiver mesmo a divergir, as condições (2.12) e (2.13) não se verificam e o processo deve ser terminado, após a execução de um número máximo de iterações permitido, n_{max}, isto é,

$$\text{terminar} \quad \text{se} \quad k \geq n_{max}.$$

O algoritmo 2.4 apresenta o critério aqui discutido.

Algoritmo 2.4 :

1. ler k, x_k, x_{k+1}, $f(x_{k+1})$, ε_1, ε_2 e calcular tol
2. se $|x_{k+1}| \leq tol$ então calcular $d = |x_{k+1} - x_k|$ senão calcular $d = \frac{|x_{k+1} - x_k|}{|x_{k+1}|}$
3. se $d \leq \varepsilon_1$ e $|f(x_{k+1})| \leq \varepsilon_2$ então terminar com convergência=.TRUE.
4. terminar com convergência=.FALSE.

A quantidade tol que aparece neste algoritmo corresponde a uma medida, muito comum, da precisão do computador. Por definição, tol é o menor número positivo, no formato de vírgula flutuante, que satisfaz $1 + tol > 1$. Este número está relacionado com o comprimento da mantissa, com a base e a aritmética do computador. Se o computador usa arredondamentos, então $tol = 0.5 \times b^{1-n}$, em que b é a base de representação dos números e n o comprimento da mantissa.

Um algoritmo que calcula o menor múltiplo de 2 que verifica $1 + tol > 1$ é o seguinte:

Algoritmo 2.5 :

1. fazer $tol = 1$
2. fazer $tol = \frac{tol}{2}$
3. fazer $t = 1 + tol$
4. se $t > 1$ então ir para o passo 2
5. terminar com $tol = tol + tol$.

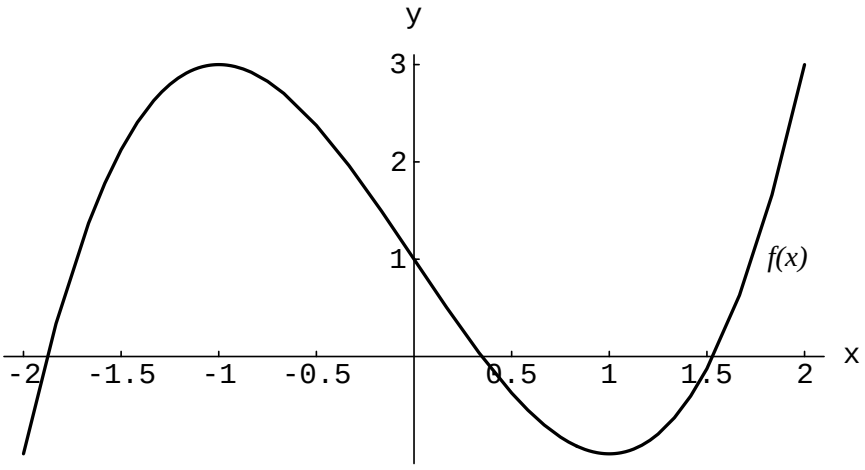


Figura 2.2: Gráfico da função $x^3 - 3x + 1$

2.4 Localização das raízes

2.4.1 Métodos gráficos

A localização das raízes de uma equação consiste em determinar tantos intervalos disjuntos quantas as raízes *reais* e *distintas*, de tal modo que, no interior de cada intervalo só exista *uma raiz*. O método mais simples para localizar as raízes de $f(x) = 0$ consiste em representar graficamente a função $f(x)$ e identificar, os pontos de intersecção de $f(x)$ com o eixo dos $X's$. Esses pontos representam os valores de x para os quais $f(x) = 0$.

Exemplo 2.1 Localizar as raízes de $f(x) = x^3 - 3x + 1 = 0$.

O gráfico da figura 2.2 mostra que existem três raízes reais. Uma está no intervalo $(-2.0, -1.5)$, a segunda pertence ao intervalo $(0.0, 0.5)$ e a outra está entre 1.5 e 2.0.

Por vezes, torna-se mais vantajoso representar duas funções, que estejam relacionadas com $f(x)$, se elas tiverem expressões mais simples do que a função $f(x)$. Assim, $f(x)$ é separada em duas

$$f(x) = g(x) - h(x),$$

de tal forma que

$$\text{se} \quad f(x) = 0 \quad \text{então} \quad g(x) = h(x)$$

e as raízes reais de $f(x) = 0$ são os *pontos de intersecção* das duas funções $g(x)$ e $h(x)$.

Exemplo 2.2 Separar $f(x) = x^3 - 3x + 1$ em $g(x) - h(x) = x^3 - (3x - 1)$ e localizar as raízes de $f(x) = 0$.

Como $x^3 - (3x - 1) = 0$ é equivalente a $x^3 = 3x - 1$, na figura 2.3 temos as duas funções representadas graficamente: $g(x) = x^3$ e $h(x) = 3x - 1$. Os pontos de intersecção de g e h estão localizados nos intervalos: $(-2.0, -1.5)$, $(0.0, 0.5)$ e $(1.5, 2.0)$.

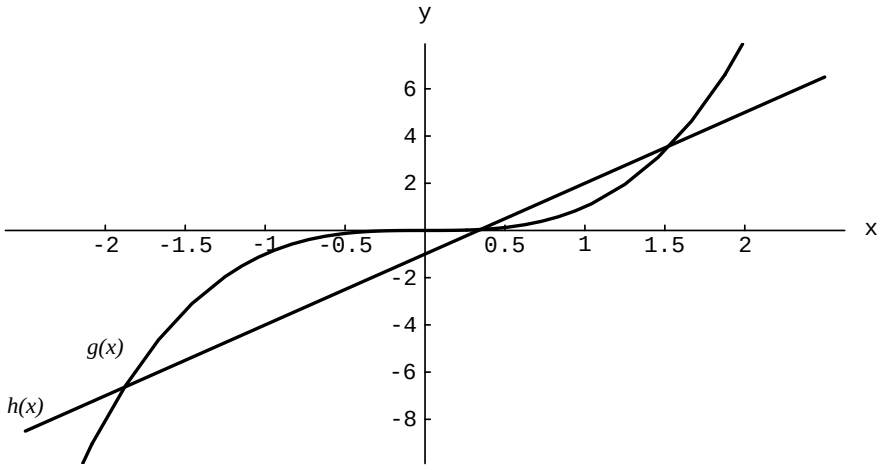


Figura 2.3: Gráficos das funções x^3 e $3x - 1$

2.4.2 Números de Rolle

Pode-se localizar as raízes de uma equação utilizando os números de Rolle.

Definição 2.1 : Os números de Rolle de uma equação $f(x) = 0$ são

i) os pontos fronteira do domínio da função $f(x)$;

ii) os zeros da primeira derivada, $f'(x)$.

Utiliza-se, também, a seguinte propriedade:

Propriedade 2.1 : Entre dois números de Rolle consecutivos, não pode existir mais do que uma raiz da equação $f(x) = 0$.

De acordo com a propriedade enunciada e se se verificar

$$f(x_i)f(x_j) < 0,$$

sendo x_i e x_j dois números de Rolle consecutivos, e uma vez que f é contínua, existe um único $x^* \in [x_i, x_j]$ tal que $f(x^*) = 0$. Na aplicação desta técnica, surgem, por vezes, algumas dificuldades. O cálculo dos zeros de $f'(x)$ é tão ou mais complicado do que o cálculo dos zeros da função original.

2.4.3 Sequências de Stürm

A aplicação do teorema de Stürm, que a seguir se enuncia, fornece um processo para localizar raízes reais de uma equação. Começaremos por definir sequência de Stürm.

Definição 2.2 : $f_0, f_1, \ldots, f_m$ *formam uma sequência de Stürm*[4] *num intervalo* $[a, b]$ *se*

i) $f_0(x)$ *tem quando muito zeros isolados em* $[a, b]$,

ii) $f_m(x) \neq 0$ *em* $[a, b]$,

iii) para qualquer valor α *do intervalo* $[a, b]$, *se* $f_j(\alpha) = 0$, *então* $f_{j-1}(\alpha)f_{j+1}(\alpha) < 0$,

iv) para qualquer valor α *do intervalo* $[a, b]$, *se* $f_0(\alpha) = 0$, *então* $f_0'(\alpha)f_1(\alpha) > 0$.

Teorema 2.1 (de Stürm) : O número de zeros da função $f_0(x)$ *num intervalo* $[a, b]$, *é igual à diferença entre o número de variações de sinal das sequências*

$$\{f_0(a), f_1(a), \ldots, f_m(a)\}$$

e

$$\{f_0(b), f_1(b), \ldots, f_m(b)\}$$

com $m \leq n$, *se* $f_0(a)f_0(b) \neq 0$ *e as funções* $f_0, f_1, \ldots, f_m$ *formarem uma sequência de Stürm.*

Iniciando o processo da localização dos zeros com o intervalo $[a, b]$, calcula-se o seu ponto médio, aplicando novamente o Teorema 2.1 aos subintervalos $[a, \frac{a+b}{2}]$ e $[\frac{a+b}{2}, b]$. O processo da divisão dos subintervalos repete-se até se obterem intervalos que contenham uma só raiz.

No caso de uma equação algébrica da forma $p_n(x) = 0$, é possível gerar um conjunto de m funções que formam uma sequência de Stürm da seguinte maneira: a primeira $f_0(x)$ é o próprio polinómio, $p_n(x)$; a segunda função, $f_1(x)$, é a primeira derivada do polinómio, $p_n'(x)$; a terceira, é o simétrico do resto da divisão de $f_0(x)$ por $f_1(x)$; a quarta, é igual ao simétrico do resto da divisão de $f_1(x)$ por $f_2(x)$; e, assim por adiante, de tal modo que as funções quociente sejam lineares em x e a última função $f_m(x)$ é o factor comum de $f_0(x)$ e $f_1(x)$ (se for *constante* então $f_0(x)$ tem zeros isolados).

2.5 Método da bissecção

2.5.1 Introdução teórica. Convergência

O *método da bissecção* é um método de encaixe. O intervalo inicial $[x_i, x_s]$ (x_i para limite inferior e x_s para limite superior) deve conter uma única raiz, para se garantir convergência. A condição

$$f(x_i)f(x_s) < 0$$

[4] C. F. Stürm nasceu na Suiça e viveu entre 1803 e 1855. Foi professor na Universidade de Sorbonne e notabilizou-se pelos seus trabalhos sobre os aspectos numéricos da Teoria das equações (Goldstine (1977)).

estabelece que, dentro deste intervalo, e se a função for contínua, existe um número impar de raízes. O método gráfico pode ajudar a identificar esse número. A aproximação à raiz é conseguida calculando o ponto médio deste intervalo,

$$x_k = \frac{x_i + x_s}{2},\ k = 1, 2, \ldots. \tag{2.14}$$

Dos subintervalos definidos $[x_i, x_k]$ e $[x_k, x_s]$, aquele que contém a raiz e deve ser escolhido para a iteração seguinte, é

$$[x_i, x_k] \text{ se } f(x_i)f(x_k) < 0 \text{ e faz-se } x_s \leftarrow x_k$$

ou

$$[x_k, x_s] \text{ se } f(x_k)f(x_s) < 0 \text{ e faz-se } x_i \leftarrow x_k.$$

O processo de divisão do intervalo ao meio e a escolha do subintervalo para a iteração seguinte, repetem-se, até o critério de paragem ser verificado, garantindo uma aproximação com uma certa precisão.

Se $[x_i, x_s]$ for o intervalo da iteração $k+1$, o intervalo da iteração $k+2$ vai ser $[x_i, x_{k+1}]$ ou $[x_{k+1}, x_s]$, com x_{k+1} definido por (2.14). Daqui se tira que

$$|x_s - x_{k+1}| = |x_{k+1} - x_i| = \frac{1}{2}|x_s - x_i|$$

ou

$$|x^* - x_{k+1}| \approx \frac{1}{2}|x^* - x_k|$$

uma vez que na iteração anterior se tinha $x_s = x_k$ (ou $x_i = x_k$) e x_i (ou x_s) $\rightarrow x^*$. Assim,

$$lim_{k\to\infty}\frac{|\epsilon_{k+1}|}{|\epsilon_k|} = \frac{1}{2}$$

sendo $\epsilon_k = x^* - x_k$, o erro da iteração k. O método da bissecção tem *convergência linear* . O intervalo da iteração k tem uma amplitude que é $2^{-(k-1)}$ vezes a amplitude do intervalo inicial.

A convergência deste método é bastante lenta e não depende de $f(x)$. Apenas o sinal de f é considerado. Usando os valores de $f(x)$, e até da derivada, obtêm-se métodos mais rápidos, como veremos mais adiante. O algoritmo que corresponde ao método da bissecção é o algoritmo 2.6.

Algoritmo 2.6 :

1. introduzir a função $f(x)$
2. ler x_i, x_s e n_{max}, calcular $f(x_i)$ e $f(x_s)$ e fazer $k = 0$
3. se $f(x_i)f(x_s) > 0$ então recomeçar com outros valores, ir para o passo 2
4. fazer $k = k + 1$

5. calcular $x_k = \frac{x_i + x_s}{2}$ e $f(x_k)$

6. se convergência=.FALSE. (algoritmo 2.4) então

   6.1. se $k \geq n_{max}$ então recomeçar, ir para o passo 2

   6.2. se $f(x_i)f(x_k) \leq 0$ então fazer $x_s = x_k$ e $f(x_s) = f(x_k)$

   6.3. senão fazer $x_i = x_k$ e $f(x_i) = f(x_k)$

   6.4. ir para o passo 4

7. terminar com $x^* \leftarrow x_k$ e $f(x^*) \leftarrow f(x_k)$.

2.5.2 Implementação. Rotina BISSEC

A implementação do algoritmo 2.6 origina a rotina BISSEC que fornece os seguintes resultados. Os valores de ε_1 e ε_2 usados no critério de paragem (algoritmo 2.4) são 10^{-6}.

Exemplo 2.3 No cálculo da raiz de $cos(x) - cos(3.1x) = 0$ e a partir do intervalo $[-1, 8]$, obtém-se, ao fim de 24 iterações o valor $x = 1.532484$ com $f(x) = -0.000001$. As seis primeiras iterações, muito lentas, foram

k	x_k	$f(x_k)$	k	x_k	$f(x_k)$
1	3.5	-0.791396	5	1.531250	0.005057
2	1.25	1.058220	6	1.671875	-0.554170
3	2.375	-1.192211	8	1.566406	-0.138589
4	1.8125	-1.026622	11	1.535645	-0.012946

O gráfico da função $f(x)$ encontra-se, mais à frente, na figura 2.4. No intervalo usado existem nove raízes e o processo convergiu para a raiz do intervalo $[1, 2]$. Se quiséssemos convergir para a raiz dupla perto de 3, teríamos de usar outro método.

Exemplo 2.4 A partir do intervalo $[7, 10]$, obtém-se, para uma das raízes da equação polinomial $f(x) = x^4 - 3x^2 + 75x - 10000 = 0$, o valor 9.886003, com $f(x) = -0.000000$ tendo o processo levado 33 iterações. No intervalo $[0, 1]$ não existe qualquer raiz.

2.6 Método da falsa posição

2.6.1 Introdução teórica. Convergência

O *método da falsa posição* pertence à classe dos métodos de encaixe. As duas aproximações iniciais à solução devem ser escolhidas por forma a

$$f(x_i)f(x_s) < 0$$

e o intervalo definido por elas, $[x_i, x_s]$, deve conter apenas a raiz procurada. A equação iterativa que gera a sequência $\{x_k\}$ de aproximações à solução, é a seguinte,

$$x_k = x_s - \frac{(x_s - x_i)f(x_s)}{f(x_s) - f(x_i)}, \quad k = 1, 2, \ldots \tag{2.15}$$

e corresponde a aproximar, na vizinhança da raiz, a função $f(x)$ por uma recta que passa pelos dois pontos, $(x_i, f(x_i))$ e $(x_s, f(x_s))$ e calcular a raiz da equação linear, isto é, o ponto de intersecção da recta com o eixo dos $X's$.

Como o intervalo para a iteração seguinte deve conter a raiz procurada, dos subintervalos definidos $[x_i, x_k]$ e $[x_k, x_s]$ escolhe-se:

$$[x_i, x_k] \text{ se } f(x_i)f(x_k) < 0 \text{ e faz-se } x_s \leftarrow x_k$$

ou

$$[x_k, x_s] \text{ se } f(x_k)f(x_s) < 0 \text{ e faz-se } x_i \leftarrow x_k.$$

O método da falsa posição, tal como o método da bissecção, converge sempre, desde que a função seja contínua. A ordem de convergência é de 1.618, igual à do método da secante (veja-se em 2.7) e a correspondente dedução é apresentada em 2.7.1. Contudo, em certas situações, é um método de primeira ordem. Quando a função é convexa no intervalo inicial $[x_i, x_s]$, o ponto x_i inicial mantém-se ao longo das iterações e se

$$\epsilon_k = x^* - x_k$$

for o erro da aproximação x_k, obtida na iteração k, da relação (2.20) (mais adiante)

$$|\epsilon_{k+1}| \approx K|\epsilon_k||\epsilon_{k-1}| \text{ tem-se com } |\epsilon_{k-1}| \approx |\epsilon_i|,$$

$$|\epsilon_{k+1}| \approx K|\epsilon_k||\epsilon_i| = K_1|\epsilon_k|,$$

com K diferente de zero e $K_1 = K|\epsilon_i|$ constantes, e

$$lim_{k\to\infty}\frac{|\epsilon_{k+1}|}{|\epsilon_k|} = K_1$$

que estabelece a ordem *um* de convergência. O método é robusto no princípio das iterações, especialmente se nos encontrarmos longe da solução, mas torna-se muito lento perto da solução. O algoritmo que corresponde ao método da falsa posição está representado no algoritmo 2.7.

Algoritmo 2.7 :

1. introduzir a função $f(x)$
2. ler x_i, x_s e n_{max}, calcular $f(x_i)$ e $f(x_s)$ e fazer $k = 0$
3. se $f(x_i)f(x_s) > 0$ então recomeçar com outros valores, ir para o passo 2
4. fazer $k = k + 1$
5. calcular $x_k = x_s - f(x_s)\frac{x_s - x_i}{f(x_s) - f(x_i)}$ e $f(x_k)$
6. se convergência=.FALSE. (algoritmo 2.4) então
   - 6.1. se $k \geq n_{max}$ então recomeçar, ir para o passo 2
   - 6.2. se $f(x_i)f(x_k) \leq 0$ então fazer $x_s = x_k$ e $f(x_s) = f(x_k)$
   - 6.3. senão fazer $x_i = x_k$ e $f(x_i) = f(x_k)$
   - 6.4. ir para o passo 4
7. terminar com $x^* \leftarrow x_k$ e $f(x^*) \leftarrow f(x_k)$.

2.6.2 Implementação. Rotina FALPOS

A rotina FALPOS implementa o algoritmo 2.7. A implementação é, em geral, muito mais rápida do que a da rotina BISSEC.

Exemplo 2.5 A partir do intervalo $[-1, 8]$, FALPOS calcula uma solução da equação do exemplo 2.3, ao fim de 9 iterações. O valor obtido foi $x = 1.532484$ com $f(x) = -0.000000$. Este processo é mais rápido do que o da bissecção, especialmente quando já estamos perto da solução:

k	x_k	$f(x_k)$	k	x_k	$f(x_k)$
1	4.267861	-1.217567	6	1.530017	0.010109
2	1.941432	-1.327367	7	1.532487	-0.000010
3	0.579511	1.060508	8	1.532484	-0.000000

Exemplo 2.6 Apenas 6 iterações são necessárias para calcular a solução da equação do exemplo 2.4. O intervalo inicial foi o mesmo $[7, 10]$ e o valor obtido foi $x = 9.886003$ com $f(x) = -0.000000$.

2.7 Método da secante

2.7.1 Introdução teórica. Convergência

Para calcular a raiz da equação $f(x) = 0$, o *método da secante* baseia-se na aproximação de $f(x)$ por uma recta, na vizinhança da raiz. O ponto de intersecção da recta com o eixo dos $X's$ é considerado como aproximação à raiz de $f(x) = 0$. Se ainda estivermos longe da solução x^*, o processo deve ser repetido iterativamente.

Para iniciar o processo iterativo são escolhidos dois pontos, x_1 e x_2. O intervalo definido por eles não necessita de conter a raiz. O ponto de intersecção da recta, que passa pelos dois pontos, com o eixo dos $X's$ obtém-se a partir da equação iterativa, que na sua forma geral é,

$$x_{k+1} = x_k - \frac{(x_k - x_{k-1}) f(x_k)}{f(x_k) - f(x_{k-1})}, \quad k = 2, 3, \ldots \tag{2.16}$$

Embora sejam necessários dois pontos para iniciar o processo iterativo, apenas um novo ponto e o correspondente valor da função são calculados em cada iteração.

Uma vez que podem ocorrer situações de 'overflow', se numa iteração $f(x_k) \approx f(x_{k-1})$, aconselha-se a implementação da seguinte fórmula

$$x_{k+1} = x_k - \frac{(x_{k-1} - x_k) \frac{f(x_k)}{f(x_{k-1})}}{(1 - \frac{f(x_k)}{f(x_{k-1})})}, \tag{2.17}$$

em vez de (2.16), caso o valor de $|f(x_{k-1})|$ seja superior a $|f(x_k)|$; senão, trocam-se os valores de x_k e x_{k-1}, bem como os correspondentes valores da função, antes de utilizar (2.17). O algoritmo 2.8 representa o algoritmo que corresponde ao método da secante.

Algoritmo 2.8 :

1. introduzir a função $f(x)$ e calcular tol
2. ler x_1, x_2 e n_{max}, calcular $f(x_1)$ e $f(x_2)$ e fazer $k = 1$
3. fazer $k = k + 1$
4. calcular $den = f(x_k) - f(x_{k-1})$
5. se $|den| > tol$ então calcular $x_{k+1} = x_k - f(x_k)\frac{(x_k - x_{k-1})}{den}$ e $f(x_{k+1})$ senão
   5.1. se $|f(x_{k-1})| < |f(x_k)|$ então trocar x_k com x_{k-1} e $f(x_k)$ com $f(x_{k-1})$
   5.2. calcular $x_{k+1} = x_k - \frac{(x_{k-1} - x_k)\frac{f(x_k)}{f(x_{k-1})}}{(1 - \frac{f(x_k)}{f(x_{k-1})})}$ e $f(x_{k+1})$
6. se convergência=.FALSE. (algoritmo 2.4) então
   6.1. se $k > n_{max}$ então recomeçar, ir para o passo 2
   6.2. ir para o passo 3
7. terminar com $x^* \leftarrow x_{k+1}$ e $f(x^*) \leftarrow f(x_{k+1})$.

Analisemos, em seguida, a convergência do método da secante.

Proposição 2.1 : Seja $f(x)$ uma função com segunda derivada contínua e suponha que o ponto x^* é tal que $f(x^*) = 0$ e $f'(x^*) \neq 0$. Então, para x_1 suficientemente perto de x^* a sequência $\{x_k\}$ gerada pelo método da secante converge para x^*, sendo a ordem de convergência igual a 1.618.

De acordo com a fórmula interpoladora de Newton, baseada em diferenças divididas (Capítulo 6, em 6.3.2), tem-se

$$f(x) = f(x_k) + (x - x_k)[x_{k-1}, x_k] + \frac{1}{2}(x - x_{k-1})(x - x_k)f''(\xi) \tag{2.18}$$

em que $[x_{k-1}, x_k]$ é a diferença dividida de primeira ordem definida por

$$[x_{k-1}, x_k] = \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}$$

e ξ é um ponto do intervalo definido pelos três pontos x, x_{k-1} e x_k. Substituindo x por x^* em (2.18) e somando a equação (2.16), agora escrita na forma

$$x_{k+1}[x_{k-1}, x_k] = x_k[x_{k-1}, x_k] - f(x_k),$$

obtém-se

$$(x^* - x_{k+1})[x_{k-1}, x_k] = -\frac{1}{2}(x^* - x_{k-1})(x^* - x_k)f''(\xi).$$

De acordo com o teorema do valor médio[5], tem-se

$$[x_{k-1}, x_k] = f'(\eta), \;\; \eta \in [x_{k-1}, x_k]$$

e considerando o erro da iteração k como sendo $\epsilon_k = x^* - x_k$, obtém-se a seguinte relação

$$\epsilon_{k+1} = -\frac{f''(\xi)}{2f'(\eta)}\epsilon_{k-1}\epsilon_k. \tag{2.19}$$

Quando $k \to \infty$, temos $\xi \approx x^*$ e $\eta \approx x^*$, e (2.19) reduz-se a

$$|\epsilon_{k+1}| \approx K|\epsilon_{k-1}||\epsilon_k| \tag{2.20}$$

em que K é uma constante diferente de zero. Considerando $|\epsilon_{k+1}| \approx C|\epsilon_k|^p$, $|\epsilon_k| \approx C|\epsilon_{k-1}|^p$ e substituindo em (2.20) chega-se a

$$C|\epsilon_k|^p \approx KC^{-\frac{1}{p}}|\epsilon_k|^{\frac{1}{p}}|\epsilon_k|.$$

Esta relação pode ser verificada se $p = 1 + \frac{1}{p}$ e $K = C^{\frac{1}{p}+1}$ ou ainda $K = C^p$. A raiz positiva da equação $p = 1 + \frac{1}{p}$ é 1.618, donde se conclui, que a ordem de convergência do método da secante é $p = 1.618$, uma vez que

$$|\epsilon_{k+1}| \approx K^{\frac{1}{p}}|\epsilon_k|^p$$

ou

$$lim_{k\to\infty}\frac{|\epsilon_{k+1}|}{|\epsilon_k|^p} = K^{\frac{1}{p}}, \;\; \text{com} \;\; p = 1.618.$$

Esta convergência designa-se por *convergência superlinear*.

O método da secante pode ser implementado para calcular *raízes complexas*. Neste caso, introduz-se a aritmética complexa nos cálculos e as aproximações iniciais para o processo iterativo devem ser valores complexos.

Quando a raiz que se pretende é múltipla, o método da secante converge linearmente. No entanto, é possível acelerar o processo iterativo introduzindo alterações. Assim, define-se uma nova função

$$u(x) = \frac{f(x)}{f'(x)}$$

e a equação $u(x) = 0$ tem uma raiz simples para $x = x^*$. Assim, para o cálculo da raiz de $u(x) = 0$ a equação iterativa modificada da secante é

$$x_{k+1} = x_k - \frac{(x_k - x_{k-1})u(x_k)}{u(x_k) - u(x_{k-1})}$$

[5]Teorema do valor médio: Seja $f(x)$ uma função contínua num intervalo fechado $[a, b]$ e diferenciável em (a, b), então existe pelo menos um ponto ξ de (a, b) de tal forma que

$$f(b) - f(a) = f'(\xi)(b - a).$$

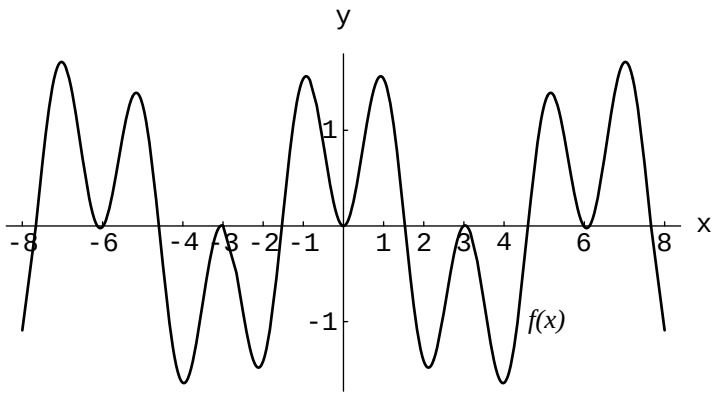


Figura 2.4: Gráfico de $f(x) = cos(x) - cos(3.1x)$

que, em termos da função original $f(x)$, tem a forma

$$x_{k+1} = x_k - \frac{(x_k - x_{k-1})f(x_k)f'(x_{k-1})}{f(x_k)f'(x_{k-1}) - f(x_{k-1})f'(x_k)},\ k = 2, 3, \ldots.$$

2.7.2 Implementação. Rotina SECANT

A rotina SECANT implementa o algoritmo 2.8. Para valores dos parâmetros $\varepsilon_1 = \varepsilon_2 = 10^{-6}$, os resultados obtidos com SECANT são:

Exemplo 2.7 Na resolução da equação do exemplo 2.3 e considerando o intervalo inicial $[-1, 8]$, obteve-se uma solução diferente da calculada nos exemplos 2.3 e 2.5. Assim, $x = 79.689180$ com $f(x) = 0.000000$, ao fim de 18 iterações.

Iniciando o processo com $[-1, 1]$, SECANT pára porque chega à situação: $f(x_{k-1}) = f(x_k)$.

Para o intervalo inicial $[0.25, 1]$, obtém-se a solução 0.000000 com $f = 0.000000$ ao fim de 37 iterações. O gráfico da função $f(x)$ está representado na figura 2.4, onde são visíveis as várias raízes.

Exemplo 2.8 Na resolução da equação $e^x - x^2 - 2x - 2 = 0$ e tomando o intervalo inicial $[-1, 0.25]$, o processo não converge e oscila entre valores positivos e negativos de x.

Também oscila entre valores positivos e negativos de x, mas acaba por convergir ao fim de 30 iterações para 2.674060, com $f = -0.000000$, quando o intervalo inicial é $[0.25, 1]$.

Converge, ao fim de 9 iterações, para $x = 2.674060$ com $f(x) = -0.000000$, quando o processo é iniciado com $[1, 2]$. A figura 2.5 apresenta o gráfico da função.

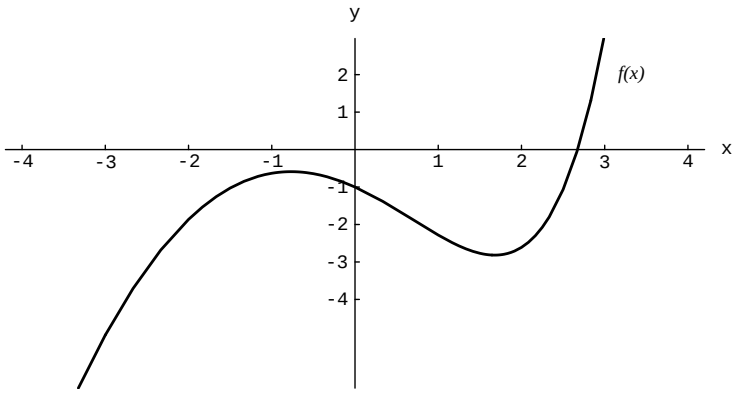


Figura 2.5: Gráfico de $f(x) = e^x - x^2 - 2x - 2$

2.8 Método de Newton-Raphson

2.8.1 Introdução teórica. Convergência

O *método de Newton*[6]*-Raphson* pode ser usado para calcular raízes reais e complexas de $f(x) = 0$. É dos métodos mais rápidos e quanto mais perto se estiver da solução mais rápida é a convergência.

[6]I. Newton nasceu no dia de Natal de 1642, no mesmo ano em que morreu Galileu. Em 1661 entrou para a Universidade de Cambridge e licenciou-se em 1665. A partir de 1667 esteve a leccionar óptica, tendo nessa altura apresentado as suas ideias sobre a origem da luz, em oposição às teorias de Descartes aceites naquela época. Newton desenvolveu um método baseado num processo iterativo de linearização, hoje conhecido por método iterativo de Newton-Raphson, numa tentativa de resolver a equação de Kepler, que publicou em 'Principia Mathematica'. Em 1669 discutiu a sua implementação na equação $x^3 - 2x - 5 = 0$. Implementação idêntica, baseada em polinómios, foi discutida e publicada por J. Raphson, em 1690, embora com a indicação de que o trabalho de Newton lhe serviu de fonte. Só a partir de 1671, com a sua invenção do telescópio reflector é que Newton passou a receber alguma atenção dos seus colegas cientistas. Recebeu, muitas vezes, críticas de outros colegas, tendo mesmo sido acusado de plágio. No período de 1675 a 1678 Newton escreveu 'Philosophiae Naturalis Principia Mathematica' onde expõe as três leis do movimento: (1) um corpo permanece em descanso a não ser que uma força actue sobre ele, (2) a força é proporcional à massa vezes a aceleração, (3) a qualquer acção corresponde uma reacção igual e de sentido oposto. Só depois deste trabalho e da publicação do livro 'Opticks' é que Newton atingiu a fama. Foi, nessa altura, eleito para Presidente da Royal Society. O interesse de Newton pelas diferenças finitas, e em particular pela interpolação, parece ter origem num desejo de ajudar um contabilista seu amigo. Alguns princípios básicos sobre a teoria da interpolação apareceram no trabalho 'Regula Differentiarum', datado por Whiteside de 1676. Um amigo de Newton, W. Jones, compilou alguns dos seus trabalhos: 'De Analysi per Aequationes Infinitas', 'De Quadratura Curvarum', 'Enumeratione' e 'Methodus Differentiallis' (um trabalho sobre interpolação baseada em diferenças centrais). Newton morreu no ano de 1727. Grande parte dos trabalhos de Newton está hoje editada em duas obras: 'Works: The Mathematical Works of Isaac Newton', dois volumes compilados por D.T. Whiteside, em 1967, e 'Papers: The Mathematical Papers of Isaac Newton', sete volumes editados também por Whiteside, durante o período de 1967 a 1976 (Goldstine (1977) e Kahaner, Moler e Nash (1989)).

A equação iterativa do método é

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}, \quad k = 1, 2, \ldots \tag{2.21}$$

sendo x_1, a aproximação inicial do processo. A equação (2.21) resulta do facto de se aproximar a função dada por uma recta, tangente a $f(x)$ no ponto da iteração corrente, x_k, e calcular o ponto de intersecção desta tangente com o eixo dos $X's$ (raiz da recta). O grande inconveniente deste método reside no facto de ser preciso calcular analiticamente a primeira derivada de $f(x)$. O algoritmo 2.9 descreve o método de Newton.

Algoritmo 2.9 :

1. introduzir as funções $f(x)$ e $f'(x)$, ler n_{max} e calcular tol
2. ler x_1 , calcular $f(x_1)$ e fazer $k = 0$
3. fazer $k = k + 1$
4. calcular $den = f'(x_k)$
5. se $|den| \leq tol$ então recomeçar com outro valor, ir para o passo 2
6. calcular $x_{k+1} = x_k - \frac{f(x_k)}{den}$ e $f(x_{k+1})$
7. se convergência=.FALSE. (algoritmo 2.4) então
   - 7.1. se $k \geq n_{max}$ então recomeçar, ir para o passo 2
   - 7.2. ir para o passo 3
8. terminar com $x^* \leftarrow x_{k+1}$ e $f(x^*) \leftarrow f(x_{k+1})$.

Veremos, em seguida, a ordem de convergência do método de Newton.

Proposição 2.2 : Seja $f(x)$ uma função com segunda derivada contínua e suponha que x^* é um ponto tal que $f(x^*) = 0$ e $f'(x^*) \neq 0$. Então, desde que a primeira aproximação à raiz, x_1, esteja suficientemente perto de x^*, a sequência $\{x_k\}$ gerada pelo método de Newton converge para x^* e a ordem de convergência é igual a dois.

Considerando a expansão em série de Taylor de $f(x)$, em relação a x_k,

$$f(x^*) = f(x_k) + (x^* - x_k)f'(x_k) + \frac{1}{2}(x^* - x_k)^2 f''(\xi), \quad \xi \in [x_k, x^*]$$

e dividindo por $f'(x_k)$, tem-se

$$0 = \frac{f(x_k)}{f'(x_k)} + (x^* - x_k) + \frac{1}{2}(x^* - x_k)^2 \frac{f''(\xi)}{f'(x_k)}$$

ou, usando a equação (2.21),

$$x^* - x_{k+1} = -\frac{1}{2}(x^* - x_k)^2 \frac{f''(\xi)}{f'(x_k)}. \tag{2.22}$$

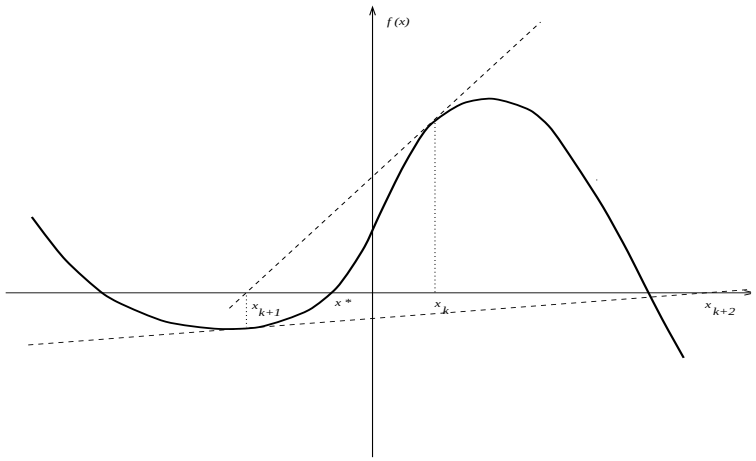


Figura 2.6: Situação de não convergência do método de Newton

Se $\epsilon_k = x^* - x_k$ for o erro da iteração k, (2.22) reduz-se a

$$\epsilon_{k+1} = -\frac{1}{2}\frac{f''(\xi)}{f'(x_k)}\epsilon_k^2, \tag{2.23}$$

e, quando $x_k \to x^*$, também $\xi \approx x^*$ e

$$lim_{k\to\infty}\frac{|\epsilon_{k+1}|}{\epsilon_k^2} = \frac{f''(x^*)}{2f'(x^*)}$$

o que define uma *convergência quadrática* para o método de Newton-Raphson.

Este tipo de convergência depende da aproximação inicial x_1, do valor de f'' em pontos da vizinhança de x^* e dos valores de $f'(x_k)$. Se estes estiverem próximos de zero, o método torna-se muito lento ou pode mesmo não convergir para a solução desejada. A figura 2.6 ilustra esta situação. Os dois teoremas seguintes fornecem condições de convergência fáceis de verificar.

Teorema 2.2 : Seja x_1 a aproximação inicial ao método de Newton e seja $\{x_k\}$ a sequência gerada por (2.21). Defina-se o intervalo

$$I = [x_1, x_1 - 2\frac{f(x_1)}{f'(x_1)}]$$

e suponha que

$$2|\frac{f(x_1)}{f'(x_1)}|\, M \leq |f'(x_1)| \quad em \quad que \quad M = max_{x\in I}|f''(x)|.$$

Então, $x_k \in I$, para $k = 2, 3, \ldots$ e

$$lim_{k\to\infty}x_k = x^*$$

sendo x^* a única raiz de $f(x) = 0$ em I.

Teorema 2.3 : Suponha que $f'(x) \neq 0$, que $f''(x)$ não muda de sinal no intervalo $[a, b]$ e que $f(a)f(b) < 0$. Se

$$|\frac{f(a)}{f'(a)}| < (b-a) \quad e \quad |\frac{f(b)}{f'(b)}| < (b-a)$$

então o método de Newton-Raphson converge a partir de uma aproximação inicial arbitrária, $x_1 \in [a, b]$.

Caso a equação não linear seja algébrica, convém alterar o algoritmo 2.9, tornando-o mais eficiente. Os valores do polinómio e da sua primeira derivada, num ponto x_k, podem ser calculados simultaneamente, usando o algoritmo 2.2. Para isto, basta introduzir o grau do polinómio, n, e os seus coeficientes $a_0, a_1, \ldots, a_n$. Assim, são as seguintes as alterações ao algoritmo.

Alterações ao algoritmo 2.9 :

1. → 1'. ler n e n_{max}, calcular *tol* e para $k = 0, 1, \ldots, n$ ler a_k
2. → 2'. ler x_1 e calcular $f(x_1) = p_n(x_1)$, $den = p'_n(x_1)$ (algoritmo 2.2) e fazer $k = 0$
3. → 3'.
4. (desaparece)
5. → 4'.
6. → 5'. calcular $x_{k+1} = x_k - \frac{f(x_k)}{den}$, $f(x_{k+1}) = p_n(x_{k+1})$ e $den = p'_n(x_{k+1})$ (algoritmo 2.2)
7. → 6'.
8. → 7'.

Para se calcularem raízes complexas procede-se como no caso do método da secante. A aproximação inicial tem de ser um valor complexo e os cálculos serão feitos em aritmética complexa.

Também se torna necessário introduzir uma alteração à equação iterativa (2.21) se a raiz procurada não é simples. Da definição da nova função

$$u(x) = \frac{f(x)}{f'(x)}$$

chega-se à equação iterativa de Newton modificada, para calcular a raiz simples de $u(x) = 0$,

$$x_{k+1} = x_k - \frac{u(x_k)}{u'(x_k)}$$

ou

$$x_{k+1} = x_k - \frac{f(x_k)f'(x_k)}{(f'(x_k))^2 - f(x_k)f''(x_k)}, \quad k = 1, 2, \ldots.$$

2.8.2 Implementação. Rotina NEWTON

Para os exemplos que a seguir se apresentam, a implementação do algoritmo 2.9, presente na rotina NEWTON, fornece os seguintes resultados:

Exemplo 2.9 Para a equação $cos(x) - cos(3.1x) = 0$, idêntica à dos exemplos 2.3, 2.5 e 2.7, a raiz calculada foi $x = -3.064968$ com $f(x) = 0.000000$, ao fim de 7 iterações. O valor inicial considerado foi $x_1 = -1$. Esta raiz é diferente das calculadas nos exemplos 2.7, 2.3 e 2.5. Veja-se o gráfico 2.4.

Se iniciarmos o processo com $x_1 = 1$, obtemos a raiz dupla 3.064968 ao fim de 7 iterações.

Com $x_1 = 1.5$, a rotina converge para 1.532484, ao fim de 3 iterações.

Exemplo 2.10 Para a equação do exemplo 2.8, obteve-se convergência para $x = 2.674060$, com $f(x) = 0.000000$, ao fim de 16 iterações, quando $x_1 = 0.25$.

No entanto, o processo levou 36 iterações a convergir para o mesmo valor, quando $x_1 = 1$. Durante 20 iterações o processo calcula valores de x do intervalo $[10, 30]$. Isto deve-se ao facto de, na primeira iteração, o valor calculado ser -0.780203, e o correspondente $f'(x) \approx 0$, o que faz com que o processo se afaste da raiz procurada.

Quando o valor inicial introduzido foi $x_1 = -1$, o processo ainda não tinha convergido ao fim de 50 iterações. Veja-se o gráfico 2.5.

Exemplo 2.11 Considerando a equação polinomial dos exemplos 2.4 e 2.6, obteve-se convergência para o mesmo valor obtido anteriormente, 9.886003, com $f(x) = 0.000000$ com qualquer dos três valores iniciais:

- $x_1 = 0$, ao fim de 15 iterações,
- $x_1 = 1$, ao fim de 15 iterações,
- $x_1 = 3$, ao fim de 12 iterações.

2.9 Método de Laguerre

2.9.1 Introdução teórica. Convergência

O *método de Laguerre*[7] para a resolução das equações algébricas

$$p_n(x) = a_0 x^n + a_1 x^{n-1} + \cdots + a_{n-1} x + a_n = 0$$

traz grandes vantagens pela rapidez de convergência. De acordo com o teorema fundamental da Álgebra, esta equação tem exactamente n raízes. Se os coeficientes do polinómio forem reais, podem aparecer pares de raízes complexas conjugadas.

A equação iterativa é definida por

$$x_{k+1} = x_k - \frac{n p_n(x_k)}{p'_n(x_k) \pm \sqrt{H(x_k)}}, \quad k = 1, 2, \ldots \tag{2.24}$$

[7] E. N. Laguerre foi um matemático francês que viveu entre 1834 e 1886. Grande parte da sua actividade profissional, de investigação e de ensino, foi desenvolvida na École Polytechnique, em Paris, onde trabalhou em domínios hoje conhecidos por Geometria Analítica, Análise, Quadratura Numérica e Equações Diferenciais (Kahaner, Moler e Nash (1989)).

em que

$$H(x_k) = (n-1)[(n-1)(p_n'(x_k))^2 - np_n(x_k)p_n''(x_k)],$$

n é o grau do polinómio, e iniciando o processo iterativo com x_1, o método gera duas sequências de aproximações que convergem para as raízes que se encontram mais próximas de x_1; uma, com um valor superior a x_1 e a outra, com um valor inferior a x_1.

Se usarmos o sinal em (2.24) que origina o menor incremento a adicionar a x_k para obter o x_{k+1}, a sequência resultante converge para a raiz mais próxima do valor inicial atribuído. Para equações algébricas com raízes reais, se x_1 estiver à esquerda da menor das raízes ou à direita da maior delas, então o processo iterativo converge respectivamente para a menor ou para a maior das raízes.

A convergência do método de Laguerre é global, pois não depende da proximidade do valor inicial atribuído na primeira iteração. Não é contudo verdade que, para raízes complexas, o método convirja para qualquer valor inicial.

Além do cálculo de $p_n(x)$, num novo ponto, em cada iteração, o método de Laguerre necessita das derivadas, $p_n'(x)$ e $p_n''(x)$. Os valores destes três polinómios podem ser calculados eficientemente por um processo recursivo, usando o algoritmo 2.3. Mesmo assim, é dos métodos que mais operações aritméticas precisa. Este inconveniente é ultrapassado pela rapidez de convergência, pois trata-se de um método de ordem de convergência igual a *três*, se a raiz procurada for simples, real ou mesmo complexa.

Para convergir para uma raiz complexa, o processo pode ser iniciado com uma aproximação inicial real, isto é, é possível que x_{k+1} seja complexo mesmo que x_k seja real, uma vez que pode-se obter $H(x_k) < 0$. Se esta situação ocorrer, deve ser introduzida aritmética complexa a partir daqui.

O algoritmo que corresponde ao método de Laguerre é o algoritmo 2.10.

Algoritmo 2.10 :

1. ler n e n_{max}, calcular tol e para $k = 0, 1, \ldots, n$ ler a_k
2. ler x_1 (real) e calcular $p_n(x_1), p_n'(x_1)$ e $p_n''(x_1)$ (algoritmo 2.3) e fazer $k = 0$
3. fazer $k = k + 1$
4. calcular $H(x_k) = (n-1)[(n-1)(p_n'(x_k))^2 - np_n(x_k)p_n''(x_k)]$
5. se $H(x_k) < 0$ então usar aritmética complexa senão usar aritmética real
6. calcular $den1 = p_n'(x_k) + \sqrt{H(x_k)}$ e $den2 = p_n'(x_k) - \sqrt{H(x_k)}$
7. se $|den1| > |den2|$ então fazer $den = den1$ senão fazer $den = den2$
8. se $|den| \leq tol$ então recomeçar, ir para o passo 2
9. calcular $x_{k+1} = x_k - \frac{np_n(x_k)}{den}$, $p_n(x_{k+1}), p_n'(x_{k+1})$ e $p_n''(x_{k+1})$ (algoritmo 2.3)
10. se convergência=.FALSE. (algoritmo 2.4) então
    - 10.1. se $k \geq n_{max}$ então recomeçar, ir para o passo 2
    - 10.2. ir para o passo 3
11. terminar com $x^* \leftarrow x_{k+1}$ e $p_n(x^*) \leftarrow p_n(x_{k+1})$.

2.9.2 Implementação. Rotina LAGUER

A rotina que implementa o algoritmo 2.10 chama-se LAGUER. Os valores dos parâmetros usados no critério de paragem são $\varepsilon_1 = \varepsilon_2 = 10^{-6}$.

Exemplo 2.12 Na resolução de $x^3 - 4x^2 + 7x - 4 = 0$ a rotina LAGUER calcula a raiz real igual a 1 com $f(x) = 0.000000$, ao fim de 4 iterações, para $x_1 = 0$.

Se o processo iterativo for iniciado com $x_1 = 3$, logo na primeira iteração $H(x_1) < 0$, e a partir daqui foi introduzida a aritmética complexa. A aproximação calculada na $1^{a.}$ iteração foi $x_2 = 1.666667 - 1.192570\imath$, em que $\imath$ é a unidade imaginária, e na quarta iteração tem-se a solução $1.5 - 1.322876\imath$ com $f(x) = 0.000000 + 0.000000\imath$.

A outra raiz da equação, conjugada desta, obteve-se ao fim de 2 iterações, tendo sido usado $x_1 = 2\imath$.

Exemplo 2.13 O polinómio $p_4(x) = x^4 + 8x^3 - 8x^2 - 200x - 425$ tem dois zeros complexos conjugados e dois reais. Iniciando o processo por $x_1 = -6$ obteve-se, ao fim de 3 iterações, o zero igual a -5.

Para $x_1 = 6$, o processo convergiu para a raiz igual a 5 e levou 2 iterações.

Quando o processo foi iniciado com $x_1 = 0$ os valores obtidos ao longo de algumas iterações foram os seguintes:

k	x_{k+1}	$f(x_{k+1})$
1	-2.336332	-73.628471
2	-4.168869	-7.837843
3	-4.614611 - 1.459362 $\imath$	-19.773438 - 20.595837 $\imath$
4	-3.957663 - 0.948867 $\imath$	-0.435078 + 1.584398 $\imath$
5	-4.000024 - 1.000024 $\imath$	+0.000096 - 0.000872 $\imath$
6	-4.000000 - 1.000000 $\imath$	+0.000000 + 0.000000 $\imath$

Na sexta iteração o critério de paragem foi verificado.

Para $x_1 = \imath$, LAGUER converge para $-4.000000 + 1.000000\imath$ ao fim de 5 iterações. Obtém-se a mesma raiz se o processo for iniciado com $x_1 = 2\imath$.

Seis iterações demora o processo a convergir para a raiz $-4.000000+1.000000\imath$, se a aproximação inicial for $x_1 = 10\imath$.

2.10 Método do ponto fixo

2.10.1 Introdução teórica. Convergência

Dada a função $f(x)$, se for possível construir uma função auxiliar, $F(x)$, de tal forma que

$$x^* = F(x^*) \quad \text{sempre} \quad \text{que} \quad f(x^*) = 0,$$

o problema do cálculo da raiz de $f(x) = 0$ reduz-se à determinação do *ponto fixo* de $F(x)$. A construção de $F(x)$ não é única. A função $F(x)$ tem um ponto fixo, num intervalo $[a, b]$, se o gráfico de $F(x)$ intersecta a recta $y = x$. Veja-se a figura 2.7 com $F(x) = cos(x) + 1$, sendo $f(x) = x - 1 - cos(x)$.

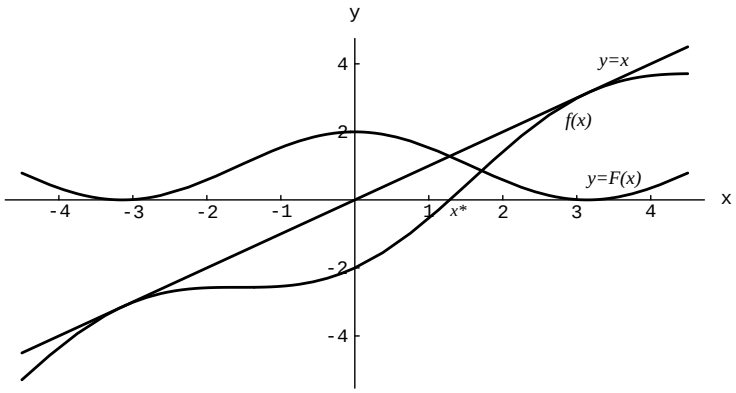


Figura 2.7: Intersecção das curvas $y = x$ e $y = F(x) = cos(x) + 1$

$F(x)$ pode ter muitos pontos fixos ou mesmo nenhum, naquele intervalo $I = [a, b]$. Para garantirmos a existência de um ponto fixo em I temos de impôr certas condições a $F(x)$.

Teorema 2.4 : Se para todo o $x \in I$, se tem $F(x) \in I$ e se $F(x)$ é uma função contínua, então $F(x)$ tem, pelo menos, um ponto fixo em I.

Para se assegurar a existência de um *único* ponto fixo, $F(x)$ não pode variar muito rapidamente em I. É preciso introduzir uma condição adicional:

Teorema 2.5 : Se para todo o $x \in I$ se tem $F(x) \in I$ e

$$|F'(x)| \leq L < 1$$

então existe exactamente um único valor x^* para o qual $x^* = F(x^*)$.

Suponha que existem dois pontos fixos, $\bar{x}$ e $\hat{x}$, ambos de I e $\bar{x} \neq \hat{x}$. De $\bar{x} = F(\bar{x})$ e $\hat{x} = F(\hat{x})$ tem-se

$$|\bar{x} - \hat{x}| = |F(\bar{x}) - F(\hat{x})| = |F'(\xi)(\bar{x} - \hat{x})| \leq L|\bar{x} - \hat{x}|$$

usando o teorema do valor médio, para ξ entre $\bar{x}$ e $\hat{x}$, ou seja

$$|\bar{x} - \hat{x}| < |\bar{x} - \hat{x}|,$$

o que é uma contradição. Ficou provado o Teorema 2.5.

A equação iterativa resultante para gerar a sequência de aproximações ao ponto fixo de $F(x)$ (ou raiz de $f(x) = 0$) é a seguinte,

$$x_{k+1} = F(x_k), \;\; k = 1, 2, \ldots \tag{2.25}$$

definindo o *método do ponto fixo*. A figura 2.8 ilustra graficamente três iterações deste processo iterativo.

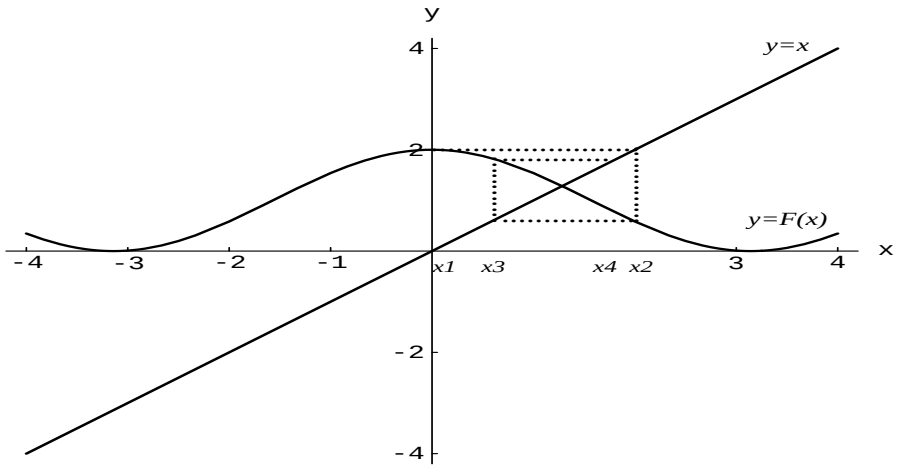


Figura 2.8: Três iterações do método do ponto fixo

Veremos, agora, a ordem de convergência da equação iterativa (2.25).

Teorema 2.6 : Se para todo o $x \in I$ *se tem* $F(x) \in I$ *e*

$$|F'(x)| \leq L < 1,$$

então para $x_1 \in I$ *a sequência* $\{x_k\}$ *gerada por (2.25) converge e a ordem de convergência é igual a um.*

Como $x^* = F(x^*)$ e $x_{k+1} = F(x_k)$, tem-se

$$x^* - x_{k+1} = F(x^*) - F(x_k) = F'(\xi)(x^* - x_k) \leq L(x^* - x_k)$$

usando o teorema do valor médio, para $\xi \in [x^*, x_k]$. Definindo $\epsilon_k = x^* - x_k$ como o erro da aproximação k, a desigualdade anterior pode escrever-se

$$|\epsilon_{k+1}| \leq L|\epsilon_k|$$

ou

$$lim_{k\to\infty} \frac{|\epsilon_{k+1}|}{|\epsilon_k|} = |F'(x^*)| < 1 \tag{2.26}$$

pois $\xi \approx x^*$, donde se conclui que a ordem de convergência é igual a *um*.

Se, por sua vez, $F'(x^*) = 0$ e $F''(x) \neq 0$ para todo o $x \in I$, obtém-se uma implementação mais rápida. A convergência passa a ser de ordem *dois*, pois

$$lim_{k\to\infty} \frac{|\epsilon_{k+1}|}{\epsilon_k^2} = \frac{|F''(x^*)|}{2}.$$

Também, se $F'(x^*) = F''(x^*) = \ldots = F^{(n)}(x^*) = 0$ e a $F^{(n+1)}(x)$ não se anula no intervalo I, então esta implementação tem convergência de ordem $n+1$:

$$lim_{k\to\infty}\frac{|\epsilon_{k+1}|}{|\epsilon_k|^{n+1}} = \frac{|F^{(n+1)}(x^*)|}{(n+1)!}.$$

Quanto mais derivadas de $F(x)$ se anularem no ponto $x = x^*$, mais rápida é a convergência do método do ponto fixo.

2.10.2 Aceleração de Aitken

Um dos processos mais conhecidos para acelerar a convergência do método do ponto fixo é o *processo* Δ^2 - *Aitken*. Se $\{x_k\}$ for uma sequência de aproximações que converge para x^*, o processo Δ^2 - Aitken gera uma nova sequência $\{y_k\}$, a partir da equação iterativa:

$$y_k = x_k - \frac{(\Delta x_k)^2}{\Delta^2 x_k},\ k = 1, 2, \ldots \tag{2.27}$$

em que

$$\Delta x_k = x_{k+1} - x_k \quad \text{e} \quad \Delta^2 x_k = x_{k+2} - 2x_{k+1} + x_k.$$

Quando a sequência $\{x_k\}$ tem convergência *regular*, tomam-se três elementos da sequência e 'extrapola-se' para o limite. A maneira como este processo acelera a convergência é estabelecida pelo seguinte teorema.

Teorema 2.7 : Sejam $\{x_k\}$ e $\{y_k\}$ duas sequências que verificam a equação (2.27), em que

$$lim_{k\to\infty} x_k = x^*.$$

Suponha ainda que $\epsilon_k = x^* - x_k$, para todo o k, satisfaz

$$\epsilon_{k+1} = (B + \beta_k)\epsilon_k, \tag{2.28}$$

com $\epsilon_k \neq 0$,

$$|B| < 1 \quad e \quad lim_{k\to\infty}\beta_k = 0.$$

Então, para k suficientemente grande, a sequência $\{y_k\}$ é bem definida e converge mais rapidamente para x^* do que a sequência $\{x_k\}$, isto é,

$$lim_{k\to\infty}\frac{|x^* - y_k|}{|x^* - x_k|} = 0.$$

A sequência $\{x_k\}$ pode ser identificada com a obtida por um método que gera iterativamente aproximações com convergência dita *regular*. Este tipo de convergência é caracterizado pela condição (2.28) apresentada no Teorema 2.7. A convergência do método do ponto fixo é disto um exemplo.

O algoritmo correspondente ao método do ponto fixo com a opção de utilização da aceleração Δ^2 - Aitken, através do parâmetro *selec*, está representado no algoritmo 2.11.

Algoritmo 2.11 :

1. introduzir a função $f(x)$, calcular tol e ler $selec$ ("Aitken"ou "ponfix") e n_{max}
2. introduzir as funções $F(x)$ e $F'(x)$
3. ler x_1, calcular $f(x_1)$ e fazer $y_0 = x_1$ e $k = 0$
4. fazer $k = k + 1$
5. calcular $deriv = F'(x_k)$
6. se $|deriv| \geq 1$ então recomeçar e ir para o passo 3 ou reintroduzir outra função $F(x)$ e ir para o passo 2
7. calcular $x_{k+1} = F(x_k)$ e $f(x_{k+1})$
8. se $k \geq 2$ e $selec =$"Aitken"então
   - 8.1. calcular $\Delta = x_k - x_{k-1}$ e $\Delta^2 = x_{k+1} - 2x_k + x_{k-1}$
   - 8.2. se $|\Delta^2| \leq tol$ então não é possível esta aceleração, ir para o passo 9
   - 8.3. calcular $y_{k-1} = x_{k-1} - \frac{(\Delta)^2}{\Delta^2}$ e $f(y_{k-1})$
9. se convergência=.FALSE. (algoritmo 2.4) então
   - 9.1. se $k \geq n_{max}$ então recomeçar, ir para o passo 3
   - 9.2. ir para o passo 4
10. terminar com $x^* \leftarrow (x_{k+1}$ ou $y_{k-1})$ e $f(x^*) \leftarrow (f(x_{k+1})$ ou $f(y_{k-1}))$.

2.10.3 Implementação. Rotina PTFIXO

A rotina PTFIXO implementa o algoritmo 2.11.

Exemplo 2.14 A equação $f(x) = x^2 - 5x + 6 = 0$ pode ser escrita na forma $x = F(x) = 5 - \frac{6}{x}$. Tomando como valor inicial $x_1 = 5$, o algoritmo 2.11 converge para a raiz $x = 3.000001$ com $f(x) = 0.000001$, ao fim de 34 iterações. No intervalo de valores de 3 a 5, $F'(x)$ toma valores entre 0.45 e 0.67, portanto, positivos e menores que um, o que define uma convergência do tipo *monótona*.

Se o processo iterativo for iniciado com $x_1 = 10$, também converge para a mesma raiz, ao fim de 34 iterações. Os valores de $F'(x)$ neste intervalo são idênticos aos do caso anterior.

Finalmente, seleccionando a aceleração Δ^2 - Aitken e tomando para x_1 o valor 5, o processo converge novamente para a raiz 3.000000 ao fim de 18 iterações. Esta formulação já não converge para valores na vizinhança de 1.

Exemplo 2.15 A equação $f(x) = x^4 - sen(x) = 0$ admite a formulação $x = F(x) = sen(x)^{\frac{1}{4}}$. Para um valor inicial, $x_1 = 2$, o método do ponto fixo baseado nesta formulação converge, ao fim de 8 iterações, para $x = 0.949617$, com $f(x) = 0.000000$.

Cinco iterações são suficientes para atingir a solução com a aceleração Δ^2 - Aitken. Os valores de $F'(x)$ naquela vizinhança rodam os 0.17 e são positivos. Tem-se uma formulação convergente e monótona.

Para $x_1 \approx 1$ ou -1, uma outra formulação, $x = F(x) = \frac{sen(x)}{x^3}$ não converge.

Exemplo 2.16 A equação algébrica $p_3(x) = x^3 - 13x + 18 = 0$ admite as seguintes formulações:

$$x = F_1(x) = \frac{13x - 18}{x^2}$$

$$x = F_2(x) = (13x - 18)^{\frac{1}{3}}$$

$$x = F_3(x) = \frac{x^3 + 18}{13}.$$

Tanto F_1 como F_2 são convergentes na vizinhança de 3. Para $x_1 = 3$, F_1 converge, embora lentamente, para a raiz $x = 2.162279$ com $f(x) = 0.000001$ e leva 47 iterações.

A aceleração Δ^2 - Aitken atinge $x = 2.162278$ ao fim de 24 iterações. A F_2 é ainda mais lenta e ao fim de 50 iterações ainda estava em 2.165661 com $f(x) = 0.003547$. A formulação F_3 não converge.

Para calcular uma raiz que esteja mais próxima de 1, a formulação F_1 não serve, pois diverge, e a F_3 converge mas muito lentamente. Ao fim de 50 iterações ainda está em 1.997695 com $f(x) = 0.002337$. Os valores de $F'(x)$ aproximam-se de 1 à medida que x se aproxima de 2.

Nem a F_1 nem a F_3 serviriam para calcular a outra raiz, que é negativa e igual a -4.162278. Quer para $x_1 = -3$ quer para $x_1 = -5$ as duas formulações são caracterizadas por valores de $F'(x)$ maiores do que um.

2.11 Problemas

1. Dado o polinómio $p_3(x) = x^3 - 4x^2 + 5x - 2$, estude o condicionamento dos zeros $x_1 = 1$ e $x_2 = 2$.

2. Use as propriedades das sequências de Stürm para localizar as raízes de $x^2 - 3x + 1 = 0$.

3. Considere a equação não linear $-1 + e^{-2x} + x = 0$.

   a) Use os números de Rolle para localizar as raízes.

   b) Calcule a raiz não nula usando o método da bissecção, com uma precisão de 10^{-6} e tomando para intervalo inicial i) $[0.0, 1.0]$; ii) $[-0.5, 1.0]$; iii) $[0.1, 1.0]$. Com qual dos intervalos obteve convergência para a raiz esperada? Justifique.

   c) Calcule a raiz mais próxima de 0.5 usando o método da falsa posição e o intervalo inicial $[0.1, 1.0]$. Comparando com os resultados obtidos em b) iii), qual das implementações foi mais rápida? Já esperava este tipo de comportamento? Justifique.

4. A equação $f(x) = x^2 + tg(x) = 0$ tem uma raiz no intervalo $[1, 2]$.

   a) Use o método da secante para a calcular.

   Pare o processo iterativo quando as condições do critério de paragem forem verificadas para $\varepsilon_1 = 0.01$ e $\varepsilon_2 = 0.05$.

b) Estude a convergência da implementação da alínea anterior.

5. Dada a equação $x + ln(x) = 0$,

   a) Use o método da bissecção para calcular a raiz que está mais próxima de 0.5, com uma precisão de 10^{-5}. Tome para intervalo inicial $[0.1, 1.0]$. Repita com os intervalos iniciais $[0.1, 3.0]$ e $[-0.5, 1.0]$.

   b) Use o método da secante, para calcular a raiz mais próxima de 0.5. Inicie o processo iterativo com o intervalo $[0.1, 1.0]$. Repita, considerando agora o intervalo inicial $[1, 2]$. Verifique que obteve a solução 0.567143, idêntica à da implementação anterior.

   c) Use o método de Newton-Raphson, com $x_1 = 0.5$, para calcular uma raiz, com precisão de 10^{-5}. Repita o processo, tomando agora $x_1 = 1.0$. O mesmo para $x_1 = 2.0$ e $x_1 = 3.0$. Que conclusões pode tirar destas implementações?

   d) Para implementar o método do ponto fixo, considere as seguintes equações iterativas:

$$i)\ x_{k+1} = e^{-x_k}$$

$$ii)\ x_{k+1} = -ln(x_k)$$

$$iii)\ x_{k+1} = \frac{x_k + e^{-x_k}}{2}$$

   Qual delas irá convergir mais rapidamente para a raiz mais próxima de 0.5? Justifique. Usando a equação iterativa seleccionada, calcule a raiz com um erro relativo inferior a 10^{-5}.

6. Dada a equação $f(x) = x - e^{-x} = 0$, calcule a raiz mais próxima de 0.55, usando o método de Newton. Pare o processo iterativo quando as condições do critério de paragem forem verificadas para $\varepsilon_1 = \varepsilon_2 = 0.0001$.

   Estude a convergência desta implementação.

7. Considere o polinómio $p_2(x) = (x - 2)x + 1$.

   a) Use as propriedades das sequências de Stürm para localizar os seus zeros.

   b) Que conclusões pode tirar da aplicação dos números de Rolle na localização dos zeros?

   c) Implemente o método da secante, a partir do intervalo $[0.5, 0.75]$, para calcular o zero do polinómio mais próximo de 0.75. Use $\varepsilon_1 = 10^{-4}$ e $\varepsilon_2 = 10^{-6}$.

   d) Estude a convergência da implementação do método de Newton-Raphson para a determinação da raiz de $p_2(x) = 0$ que está mais próxima de 0.75. Implemente o método de Newton para calcular essa raiz, com uma precisão de 10^{-4}.

   e) Implemente o método de Newton modificado para calcular a mesma raiz de d). Comente os resultados, comparando-os em termos de rapidez de convergência.

8. A função $f(x) = sen(15x) - 0.5sen(14x)$ intersecta a parte positiva do eixo dos $X's$ numa infinidade de pontos.

   a) Use o método da falsa posição para calcular os cinco primeiros zeros de $f(x)$. Use os intervalos iniciais: $[\pi/30, \pi/15], [\pi/15, 2\pi/15], [2\pi/15, 3\pi/15], [3\pi/15, 4\pi/15]$ e $[4\pi/15, 5\pi/15]$.

   b) Use o método de Newton-Raphson para calcular a menor raiz positiva de $f(x) = 0$. Tome $x_1 = \pi/15$. Se considerar $x_1 = \pi/30$, para que valor vai convergir a implementação? Repita o processo iterativo tomando para x_1 os seguintes valores: $2\pi/15$, $3\pi/15$ e $4\pi/15$. Como explica o comportamento estranho da implementação iniciada com $x_1 = \pi/30$?

9. Considere a equação $f(x) = 3x^2 + tg(x) = 0$. No intervalo $[-2, 2]$ tem quatro raízes e uma delas está próxima de 1.7. Tem dois pontos de descontinuidade em $-\pi/2$ e $\pi/2$.

   a) Use o método da secante para calcular a raiz negativa mais próxima de zero. Tome para intervalo inicial $[1.0, 2.0]$ e use uma precisão de 10^{-5}. Repita o processo com os seguintes intervalos iniciais: $[-0.25, 0.25]$, $[-0.5, 0.5]$, $[-2.0, -0.5]$, $[-2.0, -1.0]$ e $[-2.0, -1.5]$. Justifique os comportamentos encontrados. Repita mais uma vez, mas agora tomando $[1.7, 2.0]$, para convergir para a raiz mais positiva.

   b) Considere a seguinte reformulação de $f(x) = 0$:

   $$x_{k+1} = arctg(-3x_k^2).$$

   Estude a convergência do método do ponto fixo baseado nesta equação iterativa, na vizinhança das três raízes: $x_1 = 0, x_2 = -0.347...$ e $x_3 = -1.403....$ Implemente o método, usando como valores iniciais i) $x_1 = 0.1$; ii) $x_1 = -0.25$; iii) $x_1 = -0.3$; iv) $x_1 = -0.5$; v) $x_1 = -1.0$. Comente os resultados obtidos.

10. Considere a equação algébrica $f(x) = x^3 - 2x - 5 = 0$.

    a) Implemente o método de Newton-Raphson em aritmética complexa para calcular uma das raízes complexas. Tome para valor inicial $x_1 = -1 + \imath$. Apresente o resultado com uma precisão de 10^{-4}.

    b) A partir da aproximação inicial $x_1 = -1$, implemente o método de Laguerre até obter uma das raízes complexas da equação com uma precisão de 10^{-4}.

11. Para a equação $x^2 + 1 = 0$, calcule as duas raízes complexas conjugadas, usando o método de Laguerre. Inicie o processo iterativo com $x_1 = 0$, use o sinal de (2.24) que corresponde a um denominador com maior valor absoluto e considere uma precisão de 10^{-6}. Repita o processo com $x_1 = 2\imath$.

12. Considere a equação polinomial $2x^4+20x^2+338=0$. Use o método de Laguerre para calcular uma raiz complexa. Tome para valor inicial $x_1=0$. Repita com $x_1=2\imath$. Repita agora com $x_1=3$. Quais são as quatro raízes da equação?

    N.B. Use, na expressão (2.24), o sinal que corresponde ao maior denominador em valor absoluto.

13. A equação $f(x)=e^x-4x=0$ tem duas raízes positivas. A menor pertence ao intervalo $[0,0.5]$ e a maior ao intervalo $[2,2.5]$.

    Considere as duas reformulações de $f(x)=0$:

$$\text{i)}\ \ x=F_1(x)=\frac{1}{4}e^x,$$

$$\text{ii)}\ \ x=F_2(x)=ln(4x).$$

    a) Estude a continuidade e a convergência das duas reformulações relativamente às duas raízes.

    b) Calcule as raízes, usando o método do ponto fixo baseado na reformulação mais adequada para cada uma delas.

    Pare os processos iterativos quando $\varepsilon_1=\varepsilon_2=0.0001$.

Capítulo 3

Sistemas de equações lineares

3.1 Introdução

3.1.1 Forma geral do problema

Seja

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \cdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n \end{cases}$$

um sistema de n equações lineares nas n incógnitas, $x_1, x_2, \ldots, x_n$, em que o vector $b = (b_1, b_2, \ldots, b_n)^T$ é o termo independente. Este sistema pode ser colocado na forma

$$Ax = b \tag{3.1}$$

em que a matriz $A \in \mathbf{R}^{n\times n}$ é formada pelos elementos a_{ij}, para $i = 1, \ldots, n$ e $j = 1, \ldots, n$ e chama-se *matriz dos coeficientes* do sistema.

A solução do sistema (3.1) consiste no vector $x = (x_1, \ldots, x_n)^T$ que verifica simultaneamente as n equações do sistema.

3.1.2 Tipos de problemas

Definição 3.1 : A característica de uma matriz A, $c(A)$, é o número máximo de linhas paralelas, ou colunas[1]*, linearmente independentes que existem na matriz.*

Se a característica, $c(A)$, é igual a n, o sistema (3.1) é possível e determinado, isto é, tem uma única solução. Se $c(A) < n$ então o sistema $Ax = b$ pode ser possível mas indeterminado, isto é, existe um conjunto indeterminado de soluções, ou, pode ser impossível. Neste último caso, não existe uma solução que verifique simultaneamente as n equações.

[1] Teorema : Numa matriz quadrada, o número máximo de linhas independentes é igual ao número máximo de colunas independentes e, portanto, a característica tanto pode calcular-se por linhas como por colunas.

Definição 3.2 : À matriz $(A\,|\,b)$ que se obtém ampliando A com a coluna do termo independente b chama-se matriz ampliada do sistema.

Teorema 3.1: Seja $Ax = b$ um sistema de n equações nas n incógnitas $x_1, \ldots, x_n$. Seja $(A\,|\,b)$ a matriz ampliada do sistema. A condição necessária e suficiente para que o sistema seja possível é que a característica de A seja igual à característica de $(A\,|\,b)$.

Existe ainda um conjunto de sistemas especiais, conhecidos por *sistemas homogéneos*, em que o vector b é igual ao vector nulo. Neste caso, o sistema nunca é impossível. Será determinado se $c(A) = n$ e indeterminado se $c(A) < n$.

3.1.3 Tipos de métodos

Como veremos, os métodos que calculam a solução do sistema (3.1) podem ser classificados em dois grupos. Os *métodos directos* calculam a solução *exacta* do sistema $Ax = b$ ao fim de um número finito de operações elementares, caso não ocorram erros de arredondamento. Os *métodos iterativos* definidos por uma sequência infinita de operações, determinam uma sequência de aproximações $\{x^{(k)}\}$, cujo limite, é a solução exacta x. A paragem do processo iterativo origina um erro de truncatura.

A escolha do método, para a resolução de (3.1), depende das características do problema, nomeadamente em termos das propriedades da matriz dos coeficientes e da dimensão do problema. Assim, por exemplo, se a matriz só tem elementos diferentes de zero sobre a diagonal principal e nas duas diagonais acima e abaixo da diagonal principal,

$$a_{i\,j} = 0, \text{ se } |i - j| \geq 2, \ i, j = 1, \ldots, n$$

o sistema resultante diz-se *tridiagonal* e é um caso particular de um *sistema com uma matriz em banda*. Outro exemplo de um sistema com matriz em banda é

$$a_{i\,j} = 0 \text{ se } |i - j| \geq 4, \ i, j = 1, \ldots, n,$$

com cinco diagonais diferentes de zero.

Um outro exemplo em que a matriz dos coeficientes tem muitos elementos nulos é o que caracteriza as *matrizes esparsas*. A maior parte dos elementos é igual a zero, mas as posições dos elementos diferentes de zero não definem nenhum padrão determinado.

Os métodos iterativos, de que são exemplos os métodos de Jacobi e de Gauss-Seidel (veja-se em 3.6.), devem ser usados na resolução do sistema (3.1) quando este é esparso e de grande dimensão, uma vez que se torna mais económico em termos do número de operações elementares necessárias. Um sistema com 50 equações a 50 incógnitas já pode ser considerado de dimensão razoável. No entanto, na prática, podem surgir sistemas com centenas, milhares, ou mesmo dezenas de milhares de equações.

3.1.4 Índice de algoritmos

Já a seguir é apresentada a lista dos algoritmos que foram desenvolvidos para este Capítulo. São ao todo 11.

Algoritmo 3.1 Correcção iterativa para melhorar uma solução calculada

Algoritmo 3.2 Resolução de um sistema linear pelo método directo de eliminação de Gauss

Algoritmo 3.3 Resolução de um sistema linear pelo método directo de eliminação de Gauss com pivotagem parcial

Algoritmo 3.4 Resolução de um sistema tridiagonal sem pivotagem

Algoritmo 3.5 Cálculo da inversa de uma matriz ou das matrizes J e U da decomposição J A = U

Algoritmo 3.6 Decomposição triangular L U de uma matriz

Algoritmo 3.7 Decomposição Cholesky de uma matriz simétrica e definida positiva ou resolução de um sistema com matriz dos coeficientes simétrica e definida positiva

Algoritmo 3.8 Factorização Q R de uma matriz

Algoritmo 3.9 Cálculo do determinante de uma matriz

Algoritmo 3.10 Resolução de um sistema linear pelo método iterativo de Jacobi

Algoritmo 3.11 Resolução de um sistema linear pelo método iterativo de Gauss-Seidel

Os algoritmos 3.2, 3.3, 3.4, 3.7, 3.10 e 3.11 dizem respeito à resolução de um sistema de equações, por métodos directos os quatro primeiros, por métodos iterativos os dois últimos. Para a decomposição de matrizes, têm-se os algoritmos 3.6, 3.7 e 3.8. O algoritmo 3.9 calcula o determinante de uma matriz e o 3.5 calcula a sua inversa. Finalmente o algoritmo 3.1 serve para melhorar uma solução entretanto já calculada.

A selecção inteligente do algoritmo deve ser feita tendo como base as característica do problema. Assim, por exemplo, se se pretende resolver um sistema de equações normal, o algoritmo 3.3. deve ser o escolhido; se o sistema é tridiagonal, então o algoritmo seleccionado deve ser o algoritmo 3.4. Para sistemas de grandes dimensões os métodos iterativos são os aconselhados (algoritmos 3.10 e 3.11).

3.2 Condição de um sistema

3.2.1 Introdução

Vamos, em seguida, analisar os efeitos que pequenas perturbações introduzidas nos dados do problema $Ax = b$, podem originar nos resultados, x. Consideramos três casos. No primeiro, as perturbações são apenas introduzidas no vector b, termo independente. No segundo caso, são, apenas, os elementos da matriz A que sofrem pequenas variações. No último problema, são introduzidas alterações quer em b quer em A.

A *sensibilidade da solução do problema* (3.1), relativa às variações introduzidas nos dados do problema (b, A ou b e A), mede a estabilidade matemática do problema. Assim, se os resultados não forem afectados de grandes variações, quer dizer que não são sensíveis às variações nos dados e o sistema $Ax = b$ diz-se *bem condicionado*. Por sua vez, se os resultados sofrerem grandes variações, isto é, são muito sensíveis às variações nos dados do problema, o sistema diz-se *mal condicionado*.

3.2.2 Número de condição de uma matriz

Considere-se $\|.\|$ como sendo uma das normas de vectores e a sua norma subordinada[2] de matrizes.

Na análise da solução do sistema perturbado que resulta de pequenas perturbações introduzidas no *vector independente* b, é preciso comparar as soluções exactas dos seguintes sistemas:

$$Ax = b$$

e

$$A(x + \delta x) = b + \delta b,$$

em que $x + \delta x$ é a solução exacta do sistema perturbado que resulta das variações introduzidas em b, δb. Considerando a matriz A não singular e usando o facto de que $Ax = b$, tira-se da segunda equação

$$\delta x = A^{-1}\delta b,$$

e em termos de normas

$$\|\delta x\| \leq \|A^{-1}\|\|\delta b\| \text{ e } \|b\| \leq \|A\|\|x\|.$$

Daqui se conclui que o *erro relativo que afecta o resultado* é dado por,

$$\frac{\|\delta x\|}{\|x\|} \leq (\|A\|\|A^{-1}\|)\frac{\|\delta b\|}{\|b\|}, \tag{3.2}$$

ou seja, é limitado pelo erro relativo das variações em b, e o seu valor depende da quantidade $\|A\|\|A^{-1}\|$.

Se as perturbações forem, agora, introduzidas nos elementos da *matriz dos coeficientes do sistema* A, então a comparação entre as soluções deve ser feita relativa aos sistemas:

$$Ax = b$$

e

$$(A + \delta A)(x + \delta x) = b, \tag{3.3}$$

em que $x + \delta x$ é a solução do sistema perturbado que resulta das variações introduzidas em A, δA. Da equação (3.3) e de $Ax = b$, tira-se que

$$\delta x = -A^{-1}\delta A(x + \delta x) \tag{3.4}$$

[2]Dada a norma de um vector $\|.\|$ e uma matriz A, defina-se $\|Ax\|$ para todos os vectores que verificam $\|x\| = 1$. A norma de matrizes subordinada (ou induzida) à norma de vectores é dada por

$$\|A\| = \text{máx}_{\|x\|=1}\|Ax\|.$$

ou

$$\|\delta x\| \leq \|A^{-1}\|\|\delta A\|\|x + \delta x\|.$$

Esta desigualdade pode ser posta na forma

$$\frac{\|\delta x\|}{\|x + \delta x\|} \leq (\|A\|\|A^{-1}\|)\frac{\|\delta A\|}{\|A\|} \tag{3.5}$$

e novamente tem-se o *erro relativo nos resultados* (supondo que δx é pequeno) limitado pelo erro relativo das perturbações introduzidas nos dados, desta vez a matriz A, e dependendo também da quantidade $\|A\|\|A^{-1}\|$.

É ainda possível deduzir outra expressão para o erro relativo em x. Considere-se o seguinte teorema:

Teorema 3.2 : Seja $\|.\|$ *uma norma subordinada de matrizes e* B *uma matriz que satisfaz*

$$\|B\| < 1.$$

Então a matriz $I + B$ *é não singular e*

$$\|(I + B)^{-1}\| \leq \frac{1}{1 - \|B\|}.$$

Se uma matriz da forma $I + B$ *é singular, então necessariamente* $\|B\| \geq 1$ *para qualquer norma de matrizes.*

Assim, considerando $\|\delta A\| < \|A^{-1}\|^{-1}$, relação muito provável, uma vez que se tomam as variações introduzidas em A pequenas e $\|\delta A\| \approx 0$, tem-se que

$$\|A^{-1}\delta A\| \leq \|A^{-1}\|\|\delta A\| < 1$$

e a matriz $I + A^{-1}\delta A$ é não singular. De acordo com o Teorema 3.2, tem-se ainda que

$$\|(I + A^{-1}\delta A)^{-1}\| \leq \frac{1}{1 - \|A^{-1}\delta A\|} \leq \frac{1}{1 - \|A^{-1}\|\|\delta A\|}. \tag{3.6}$$

Desenvolvendo a equação (3.3), resulta

$$Ax + \delta A\, x + A\delta x + \delta A\, \delta x = b = Ax,$$

$$A^{-1}Ax + A^{-1}\delta A\, x + A^{-1}A\delta x + A^{-1}\delta A\, \delta x = A^{-1}Ax$$

ou ainda

$$x + \delta x = (I + A^{-1}\delta A)^{-1}x.$$

Substituindo esta última equação em (3.4) e usando a desigualdade (3.6) obtém-se

$$\|\delta x\| \leq \frac{\|A^{-1}\|\|\delta A\|}{1 - \|A^{-1}\|\|\delta A\|}\|x\|$$

ou, dividindo por $\|x\|$, tem-se para o erro relativo,

$$\frac{\|\delta x\|}{\|x\|} \leq \|A\|\|A^{-1}\|\frac{\|\delta A\|}{\|A\|}(\frac{1}{1-\|A^{-1}\|\|\delta A\|}). \tag{3.7}$$

Também, neste caso, o *erro relativo* das perturbações induzidas no resultado é limitado pelo erro relativo das perturbações introduzidas nos dados do problema (matriz A) e depende de $\|A\|\|A^{-1}\|$.

O terceiro caso que pode ocorrer, corresponde a introduzir pequenas *perturbações quer no vector b quer na matriz A*. Assim, iremos analisar os resultados exactos dos seguintes problemas:

$$Ax = b$$

e

$$(A + \delta A)(x + \delta x) = b + \delta b. \tag{3.8}$$

Desenvolvendo a equação (3.8) e multiplicando à esquerda por A^{-1} obtém-se

$$\delta x(I + A^{-1}\delta A) = A^{-1}\delta b - A^{-1}\delta A\, x$$

ou ainda

$$\delta x = (I + A^{-1}\delta A)^{-1}A^{-1}(\delta b - \delta A\, x).$$

Desta igualdade e usando o Teorema 3.2 encontramos para limite superior do erro absoluto nos resultados,

$$\|\delta x\| \leq \frac{\|A^{-1}\|}{(1-\|A^{-1}\|\|\delta A\|)}(\|\delta b\| + \|\delta A\|\|x\|)$$

e para o erro relativo

$$\frac{\|\delta x\|}{\|x\|} \leq \frac{\|A^{-1}\|\|A\|}{(1-\|A^{-1}\|\|\delta A\|)}(\frac{\|\delta b\|}{\|b\|} + \frac{\|\delta A\|}{\|A\|}). \tag{3.9}$$

Também neste caso o *erro relativo nos resultados* depende de $\|A\|\|A^{-1}\|$.

Nas três possíveis situações verifica-se que o erro relativo nos resultados é limitado pelos erros relativos nos dados multiplicados por $\|A\|\|A^{-1}\|$ e isto pode significar que aquele *erro pode atingir valores muito superiores aos erros introduzidos nos dados do problema*.

Definição 3.3 : A quantidade

$$k(A) = \|A\|\|A^{-1}\|$$

chama-se número de condição da matriz A.

Como $I = A\,A^{-1}$, tem-se

$$1 = \|I\| = \|A\|\,\|A^{-1}\| \leq \|A\|\,\|A^{-1}\|$$

e $k(A) \geq 1$.

O número de condição de A mede a *sensibilidade da solução* x, do sistema $Ax = b$, em relação a pequenas variações introduzidas nos dados: vector b e matriz A. Daí ser costume designar o sistema de *bem condicionado* se o $k(A)$ atinge um valor baixo e de, provavelmente, *mal condicionado* se o $k(A)$ for elevado.

Um valor próximo de zero, em valor absoluto, do *determinante da matriz* dos coeficientes do sistema na *forma escalonada*, é indicativo de que o sistema é mal condicionado. A matriz na forma escalonada obtém-se dividindo cada linha pelo elemento de maior módulo dessa linha, em valor absoluto.

Veja-se o exemplo seguinte:

Exemplo 3.1 Considere o sistema

$$\begin{cases} 10x_1 + 7x_2 + 8x_3 + 7x_4 &= 32 \\ 7x_1 + 5x_2 + 6x_3 + 5x_4 &= 23 \\ 8x_1 + 6x_2 + 10x_3 + 9x_4 &= 33 \\ 7x_1 + 5x_2 + 9x_3 + 10x_4 &= 31 \end{cases}$$

A solução calculada pelo método directo e estável de 3.4.4 (eliminação de Gauss com pivotagem parcial) é o vector $(1, 1, 1, 1)^T$ e é exacta. Se o vector independente for alterado por quantidades, em termos absolutos, da ordem de 0.1 e ficar $b + \delta b = (32.1, 22.9, 33.1, 30.9)^T$, a solução calculada (pelo mesmo algoritmo de 3.4.4) do sistema resultante perturbado, $Ax = b + \delta b$, é agora o vector $(9.2, -12.6, 4.5, -1.1)^T$. A solução é muito sensível a pequenas variações no vector b.

A inversa da matriz dos coeficientes (veja-se em 3.4.8) é

$$A^{-1} = \begin{pmatrix} 25 & -41 & 10 & -6 \\ -41 & 68 & -17 & 10 \\ 10 & -17 & 5 & -3 \\ -6 & 10 & -3 & 2 \end{pmatrix}$$

e, como

$$\|A\|_\infty = \text{máx}(32, 23, 33, 31) = 33 \text{ e } \|A^{-1}\|_\infty = \text{máx}(82, 136, 35, 21) = 136$$

tem-se $k(A) = 4488$ (cerca de 400 vezes maior do que os elementos envolvidos em A), confirmando-se o mau condicionamento do sistema.

A matriz dos coeficientes na forma escalonada é

$$A_{esc} = \begin{pmatrix} 1 & 0.7 & 0.8 & 0.7 \\ 1 & 0.71429 & 0.85714 & 0.71429 \\ 0.8 & 0.6 & 1 & 0.9 \\ 0.7 & 0.5 & 0.9 & 1 \end{pmatrix}$$

e o seu determinante (calculado pelo método de 3.5.7) é igual a 0.000143 (embora o determinante da matriz A seja igual a um), próximo de zero como se previa.

3.3 Redução do efeito dos erros de arredondamento

3.3.1 Introdução

Já foram feitas referências aos erros de arredondamento e à sua propagação no Capítulo 1. Qualquer resultado final depende dos resultados intermédios e, em geral, quanto maiores

são os erros de arredondamento cometidos durante o processo numérico, maior é o erro de arredondamento acumulado no resultado final. Em geral, quanto maior é o número de operações a efectuar no processo, maior é a acumulação de erros de arredondamento. Para contrariar estas tendências, torna-se crucial a implementação de métodos estáveis. Estes não são tão sensíveis à propagação e acumulação dos erros de arredondamento, como os instáveis. Assim, por exemplo:

Exemplo 3.2 A solução exacta do sistema

$$\begin{cases} 0.0001x_1 + x_2 = 1.0001 \\ x_1 + x_2 = 2 \end{cases}$$

é o vector $x = (1,1)^T$. Se a resolução for feita pelo método directo de eliminação de Gauss (algoritmo 3.2 em 3.4.2) e com aritmética de três algarismos significativos, surge o sistema condensado

$$\begin{array}{rcrcl} 0.0001x_1 & + & x_2 & = & 1.00 \\ & - & 1.00 \times 10^4 x_2 & = & -1.00 \times 10^4 \end{array} ,$$

em que o multiplicador é igual a -1×10^4 e a solução é $(0,1)^T$. No entanto, se o sistema for resolvido pelo método directo de eliminação de Gauss com pivotagem parcial (em 3.4.4) e aritmética de três algarismos significativos, tem-se sucessivamente

$$\begin{array}{rcrcl} x_1 + x_2 = 2 \\ 0.0001x_1 + x_2 = 1.00 \end{array} \iff \begin{array}{rcrcl} x_1 + x_2 = 2 \\ + x_2 = 1 \end{array}$$

com multiplicador igual a -1×10^{-4}. A solução calculada é agora o vector $(1,1)^T$.

A partir deste exemplo, pode-se concluir que na primeira resolução verificou-se uma acumulação muito significativa dos erros de arredondamento, originando um resultado disparatado e na segunda, essa acumulação é reduzida, de tal forma que a solução calculada é igual à solução exacta. Assim, para a resolução de um sistema de equações lineares, o método directo que deve ser usado, com o objectivo de minimizar a acumulação dos erros de arredondamento, é o de eliminação de Gauss com pivotagem parcial.

Considere, agora o seguinte exemplo,

Exemplo 3.3 O sistema

$$\begin{cases} 2x_1 + 100000x_2 = 100002 \\ x_1 + x_2 = 2 \end{cases}$$

tem como solução exacta o vector $(1,1)^T$. Se for utilizada aritmética de três algarismos significativos, a sua resolução pelo método directo de eliminação de Gauss com pivotagem parcial, origina as seguintes etapas:

$$\begin{array}{rcrcl} 2x_1 + 1.00 \times 10^5 x_2 = 1.00 \times 10^5 \\ x_1 + x_2 = 2 \end{array} \iff \begin{array}{rcrcl} 2x_1 + 1.00 \times 10^5 x_2 = 1.00 \times 10^5 \\ - 0.5 \times 10^5 x_2 = -0.5 \times 10^5 \end{array}$$

com multiplicador igual a -0.5 e solução $(0,1)^T$.

Por sua vez, se o sistema dado for, numa primeira fase, escalonado, por forma a se obter o elemento de maior módulo, em cada linha da matriz dos coeficientes, igual a um, a resolução torna-se estável. Multiplicando a primeira linha por 1×10^{-5}, obtém-se o sistema equivalente

$$\begin{array}{rcrcl} 2\times 10^{-5}x_1 & + & x_2 & = & 1 \\ x_1 & + & x_2 & = & 2 \end{array}$$

que durante o processo de condensação transforma-se em

$$\begin{array}{rcrcl} x_1 & + & x_2 & = & 2 \\ 2\times 10^{-5}x_1 & + & x_2 & = & 1 \end{array} \iff \begin{array}{rcrcl} x_1 & + & x_2 & = & 2 \\ & + & 1.00x_2 & = & 1.00 \end{array}$$

em que o multiplicador é -2×10^{-5} e a solução é $(1,1)^T$, como se pretendia.

Daqui se conclui que o sistema deve estar na sua forma escalonada quando for utilizado o método directo e estável na sua resolução, para que a acumulação de erros de arredondamento seja mínima. O escalonamento do sistema só faz sentido quando este tiver uma matriz dos coeficientes com elementos de grandezas muito distintas.

3.3.2 Correcção iterativa

Seja $\bar{x}$ a solução do sistema $Ax = b$, calculada por um processo estável. Devido aos erros de arredondamento que se cometem durante o processo numérico, $\bar{x}$ não é a solução exacta. Para *melhorar* a aproximação calculada, reduzindo os efeitos da acumulação dos erros de arredondamento, é possível calcular (estimar) o seu erro δx, de tal forma que o novo valor $\bar{x} + \delta x$ verifique exactamente o sistema,

$$A(\bar{x} + \delta x) = b.$$

Daqui, tira-se que

$$A\bar{x} + A\delta x = b$$

ou

$$A\,\delta x = b - A\bar{x}.$$

Chamando a $r = b - A\bar{x}$ o vector *resíduo* (que será nulo se $\bar{x}$ for a solução exacta do sistema), a correcção δx, que deve ser adicionada a $\bar{x}$ para se obter uma solução melhorada, deve ser calculada através da resolução do seguinte sistema:

$$A\,\delta x = r, \tag{3.10}$$

em que a matriz dos coeficientes é A e o termo independente, o vector resíduo r, é definido por

$$r = b - A\bar{x}. \tag{3.11}$$

No cálculo de r deve ser usada aritmética de dupla precisão para que a acumulação de erros de arredondamento seja mínima.

Este processo designa-se por *correcção iterativa* e define um método iterativo, uma vez que o processo pode ser repetido até se atingir um vector δx muito pequeno.

A grandeza do vector δx, da primeira iteração, pode ser usada para se concluir sobre a condição do sistema $Ax = b$, uma vez que dá uma estimativa do erro absoluto verificado

no resultado calculado $\bar{x}$, quando pequenas variações são introduzidas nos dados, durante o processo numérico. O algoritmo 3.1 corresponde ao método da correcção iterativa.

Algoritmo 3.1 :

1. ler n, ε , calcular tol e para $i = 1, \ldots, n$ ler $\bar{x}_i$, b_i e $(a_{ik}, k = 1, \ldots, n)$
2. calcular matrizes J e U tais que: $JA = U$ (algoritmo 3.5, com $selec =$ "J e U")
3. para $i = 1, \ldots, n$ calcular $r_i = b_i - \sum_{k=1}^{n} a_{ik}\bar{x}_k$ usando 'dupla precisão'
4. resolver o sistema $U\delta x = Jr$ em ordem a δx:
   - 4.1. para $i = 1, \ldots, n$ calcular $aux_i = \sum_{k=1}^{n} j_{ik} r_k$
   - 4.2. se $|u_{nn}| \leq tol$ então terminar, a matriz é singular
   - 4.3. calcular $\delta x_n = \frac{aux_n}{u_{nn}}$
   - 4.4. para $i = n-1, \ldots, 1$ calcular $soma = \sum_{k=i+1}^{n} u_{ik}\delta x_k$ e $\delta x_i = \frac{aux_i - soma}{u_{ii}}$
5. calcular $norma = \sqrt{\sum_{i=1}^{n} \delta x_i^2}$ e para $i = 1, \ldots, n$ calcular $\bar{x}_i = \bar{x}_i + \delta x_i$
6. se $norma > \varepsilon$ então ir para o passo 3
7. terminar com $x^* \leftarrow (\bar{x}_i, i = 1, \ldots, n)$.

3.3.3 Implementação. Rotina CORITE

A implementação do algoritmo 3.1 origina a rotina CORITE. O valor do parâmetro ε usado para a paragem do processo iterativo é de 10^{-12}.

Exemplo 3.4 A solução calculada do sistema

$$\begin{cases} x_1 + \frac{1}{2}x_2 + \frac{1}{3}x_3 + \frac{1}{4}x_4 + \frac{1}{5}x_5 &= 1 \\ \frac{1}{2}x_1 + \frac{1}{3}x_2 + \frac{1}{4}x_3 + \frac{1}{5}x_4 + \frac{1}{6}x_5 &= 1 \\ \frac{1}{3}x_1 + \frac{1}{4}x_2 + \frac{1}{5}x_3 + \frac{1}{6}x_4 + \frac{1}{7}x_5 &= 1 \\ \frac{1}{4}x_1 + \frac{1}{5}x_2 + \frac{1}{6}x_3 + \frac{1}{7}x_4 + \frac{1}{8}x_5 &= 1 \\ \frac{1}{5}x_1 + \frac{1}{6}x_2 + \frac{1}{7}x_3 + \frac{1}{8}x_4 + \frac{1}{9}x_5 &= 1 \end{cases}$$

é o vector $(5.000462124, -120.008245935, 630.034584633, -1120.051256725, 630.024735118)^T$. Foi implementado o algoritmo 3.3 e os dados foram introduzidos com nove casas decimais. A rotina CORITE fornece a seguinte solução melhorada, ao fim de duas iterações (os dados foram introduzidos com quinze casas decimais):

$$x = (5.000000000, -120.000000008, 630.000000034, -1120.000000052, 630.000000025)^T.$$

3.4 Métodos directos de resolução

3.4.1 Introdução teórica

Considere o sistema de equações lineares na forma (3.1). Os métodos directos de resolução têm como objectivo transformar o sistema $Ax = b$ noutro, com uma forma mais simples, $Ux = c$, que lhe seja equivalente, no sentido de que têm as mesmas soluções. A matriz U pode ter a forma triangular superior. Esta transformação é feita tendo como base um conjunto de *operações elementares* que são executadas sobre o sistema, designadamente,

1. troca de duas linhas paralelas;

2. multiplicação de uma linha por um escalar diferente de zero;

3. substituição de uma linha pela que dela se obtém adicionando o produto de outra linha paralela por um escalar.

Quando estas operações incidem sobre linhas, esta transformação designa-se por *condensação vertical*.

Quando o sistema está na forma $Ux = c$, a sua resolução não oferece qualquer dificuldade, como se verá já a seguir.

3.4.2 Eliminação de Gauss

Dos processos mais antigos, o mais conhecido e mais simples de condensação do sistema, é o método de eliminação de Gauss[3]. A condensação é feita por forma a reduzir a matriz A à forma U, triangular superior. O processo envolve, para um sistema de n equações a n incógnitas, $n-1$ etapas. Na primeira etapa anulam-se todos os elementos de A que

[3]C. F. Gauss viveu entre 1777 e 1855. Nascido de uma família muito pobre, foi protegido do Duque de Brunswick que financiou os seus estudos e o ajudou financeiramente até ao dia da sua morte, em 1806. Gauss foi professor de astronomia na Universidade de Göttingen, tendo lá permanecido até ao fim da sua vida. Desenvolveu trabalhos nos seguintes domínios: Geometria, Análise, Astronomia, Electromagnetismo, Geodesia, Teoria dos Números, tendo publicado 'Disquisitiones Arithmeticae', e Álgebra (tendo providenciado uma demonstração rigorosa do teorema fundamental da Álgebra). Em 1833, cinco anos antes de Morse, Gauss foi o primeiro a comunicar através de telégrafo eléctrico. O seu trabalho conjunto com W. Weber em electromagnetismo valeu-lhe um monumento na cidade de Göttingen. Um dos seus últimos trabalhos em geometria diferencial deu origem à teoria da relatividade de Einstein. O conceito de estimador dos mínimos quadrados foi descoberto, em 1795, por Gauss, que na altura tinha 18 anos. Foi um dos primeiros cientistas a usar a técnica dos mínimos quadrados, embora muito antes desta invenção outros cientistas tenham tentado modelar dados. Foi Gauss quem primeiro relacionou a teoria das probabilidades com o método dos mínimos quadrados no seu trabalho 'Theoria Motus', onde, também discute o problema dos erros de arredondamento como um meio que afecta a acurácia dos cálculos. Segundo a interpretação de Goldstine (1977), Gauss deu-se conta do fenómeno da estabilidade numérica. Em 1814 Gauss apresentou o trabalho 'Methodus nova integralium valores per approximationem inveniendi' sobre o seu método de integração numérica. Também em interpolação Gauss deu a sua contribuição. É considerado como um dos maiores matemáticos de todos os tempos, comparável a Arquimedes e Newton (Kahaner, Moler e Nash (1989) e Hämmerlin e Hoffmann (1991)).

estejam abaixo da posição $(1,1)$ da diagonal principal. Assim, se a matriz ampliada do sistema $Ax = b$ for

$$\left(\begin{array}{ccccc|c} a_{11} & a_{12} & \dots & a_{1\,n-1} & a_{1\,n} & b_1 \\ a_{21} & a_{22} & \dots & a_{2\,n-1} & a_{2\,n} & b_2 \\ a_{31} & a_{32} & \dots & a_{3\,n-1} & a_{3\,n} & b_3 \\ & & \dots & & & \\ a_{n-1\,1} & a_{n-1\,2} & \dots & a_{n-1\,n-1} & a_{n-1\,n} & b_{n-1} \\ a_{n\,1} & a_{n\,2} & \dots & a_{n\,n-1} & a_{n\,n} & b_n \end{array}\right)$$

calculam-se os $n-1$ *multiplicadores* da primeira etapa

$$m_{j\,1} = -\frac{a_{j\,1}}{a_{11}}, \text{ para } j = 2, \dots, n \tag{3.12}$$

sendo o elemento a_{11} considerado o 'pivot'. A multiplicação de $m_{j\,1}$ pelos elementos da primeira linha, com a adição dos elementos da linha j, reduz $a_{j\,1}$ a zero, modificando os restantes elementos da linha j, $j = 2, \dots, n$. Ao fim desta etapa a matriz tem o seguinte aspecto

$$\left(\begin{array}{ccccc|c} a_{11} & a_{12} & \dots & a_{1\,n-1} & a_{1\,n} & b_1 \\ 0 & a'_{22} & \dots & a'_{2\,n-1} & a'_{2\,n} & b'_2 \\ 0 & a'_{32} & \dots & a'_{3\,n-1} & a'_{3\,n} & b'_3 \\ & & \dots & & & \\ 0 & a'_{n-1\,2} & \dots & a'_{n-1\,n-1} & a'_{n-1\,n} & b'_{n-1} \\ 0 & a'_{n\,2} & \dots & a'_{n\,n-1} & a'_{n\,n} & b'_n \end{array}\right)$$

em que a designação $a'_{i\,j}$ significa que esses elementos foram modificados. A segunda etapa vai anular os elementos que se encontram na segunda coluna e abaixo da posição da diagonal principal, e que são ao todo $n-2$. Os escalares, para a utilização da operação elementar 3., são os seguintes *multiplicadores*

$$m_{j\,2} = -\frac{a'_{j\,2}}{a'_{22}}, \text{ para } j = 3, \dots, n \tag{3.13}$$

sendo a'_{22} o 'pivot' desta segunda etapa. A multiplicação de $m_{j\,2}$ pelos elementos da segunda linha, com a adição dos elementos da linha j, anula o $a'_{j\,2}$ e altera os restantes elementos dessa linha j, $j = 3, \dots, n$. A matriz ampliada é transformada e tem agora o aspecto

$$\left(\begin{array}{ccccc|c} a_{11} & a_{12} & \dots & a_{1\,n-1} & a_{1\,n} & b_1 \\ 0 & a'_{22} & \dots & a'_{2\,n-1} & a'_{2\,n} & b'_2 \\ 0 & 0 & \dots & a''_{3\,n-1} & a''_{3\,n} & b''_3 \\ & & \dots & & & \\ 0 & 0 & \dots & a''_{n-1\,n-1} & a''_{n-1\,n} & b''_{n-1} \\ 0 & 0 & \dots & a''_{n\,n-1} & a''_{n\,n} & b''_n \end{array}\right)$$

indicando $a''_{i\,j}$ que o valor foi novamente alterado.

Este processo repete-se, nas etapas seguintes $k = 3, \ldots$, usando-se

$$m_{j\,k} = -\frac{a_{j\,k}}{a_{k\,k}}, \text{ para } j = k+1, \ldots, n$$

para calcular os *multiplicadores*, sempre em função dos elementos da última matriz transformada, sendo que na última etapa, $k = n-1$, só resta anular o elemento da posição $(n, n-1)$. O *multiplicador* é, então, definido por

$$m_{j\,n-1} = -\frac{a_{j\,n-1}}{a_{n-1\,n-1}}, \text{ para } j = n \tag{3.14}$$

que multiplicado pelos elementos da linha $n-1$ e adicionados aos elementos da linha n, transforma a matriz à sua forma condensada final

$$(U \mid c) = \left(\begin{array}{ccccc|c} u_{11} & u_{12} & \ldots & u_{1\,n-1} & u_{1\,n} & c_1 \\ 0 & u_{22} & \ldots & u_{2\,n-1} & u_{2\,n} & c_2 \\ 0 & 0 & \ldots & u_{3\,n-1} & u_{3\,n} & c_3 \\ & & \ldots & & & \\ 0 & 0 & & u_{n-1\,n-1} & u_{n-1\,n} & c_{n-1} \\ & & & 0 & u_{n\,n} & c_n \end{array}\right).$$

A matriz do lado esquerdo tem a forma triangular superior. O sistema resultante

$$\begin{array}{rrcrrl} u_{11}x_1 & +u_{12}x_2 & + & \cdots & +u_{1n}x_n & = c_1 \\ & u_{22}x_2 & + & \cdots & +u_{2n}x_n & = c_2 \\ & & & & \ldots & \\ & & & u_{n-1\,n-1}x_{n-1} & +u_{n-1\,n}x_n & = c_{n-1} \\ & & & & u_{nn}x_n & = c_n \end{array}$$

resolve-se facilmente por *substituição inversa*, que consiste em retirar da última equação o valor de x_n; retirar da penúltima equação o valor de x_{n-1}, utilizando o valor já calculado de x_n, e assim por adiante. Da segunda equação retira-se o x_2, substituindo os valores de $x_n, x_{n-1}, \ldots, x_3$ já calculados e finalmente da primeira equação retira-se o x_1. As equações correspondentes são:

$$x_n = \frac{c_n}{u_{n\,n}} \tag{3.15}$$

e

$$x_i = \frac{c_i - \sum_{j=i+1}^{n} u_{i\,j}x_j}{u_{i\,i}}, \text{ para } i = n-1, n-2, \ldots, 2, 1. \tag{3.16}$$

O algoritmo 3.2 corresponde a este processo de condensação de Gauss.

Algoritmo 3.2 :

1. `ler` n, `calcular` *tol* `e para` $i = 1, \ldots, n$ `ler` b_i `e` $(a_{ij}, j = 1, \ldots, n)$
2. `para` $j = 1, \ldots, n-1$ `fazer`

2.1. se $|a_{jj}| \leq tol$ então terminar sem solução

2.2. para $i = j+1, \ldots, n$ calcular $m_{ij} = -\frac{a_{ij}}{a_{jj}}$

2.2.1. para $k = j+1, \ldots, n$ calcular $a_{ik} = a_{ik} + m_{ij}a_{jk}$

2.2.2. calcular $b_i = b_i + m_{ij}b_j$

3. se $|a_{nn}| \leq tol$ então terminar sem solução

4. calcular $x_n = \frac{b_n}{a_{nn}}$

5. para $i = n-1, \ldots, 1$ calcular $soma = \sum_{j=i+1}^{n} a_{ij}x_j$ e $x_i = \frac{b_i - soma}{a_{ii}}$

6. terminar com $x^* \leftarrow (x_i, i = 1, \ldots, n)$.

3.4.3 Implementação. Rotina GAUSS

O algoritmo 3.2 foi implementado, originando a rotina GAUSS, e os resultados obtidos são:

Exemplo 3.5 Usando o método directo de eliminação de Gauss, na resolução do sistema

$$\begin{cases} 3x_1 + 2x_2 - 2x_3 &= 1 \\ 9x_1 + 7x_2 - 9x_3 &= 1 \\ 6x_1 + 8x_2 - 8x_3 &= 1 \end{cases}$$

obtêm-se os seguintes multiplicadores da primeira etapa $m_{21} = -3$ e $m_{31} = -2$. Na segunda etapa, tem-se $m_{32} = -4$. A matriz ampliada final $(U \mid c)$ é

$$\left(\begin{array}{ccc|c} 3 & 2 & -2 & 1 \\ 0 & 1 & -3 & -2 \\ 0 & 0 & 8 & 7 \end{array}\right)$$

e das equações (3.15) e (3.16) (substituição inversa) tira-se a solução $(0.5, 0.625, 0.875)^T$.

3.4.4 Eliminação de Gauss com pivotagem parcial

Os multiplicadores calculados em cada uma das etapas do processo de condensação podem chegar a atingir valores enormes se os elementos considerados 'pivot' de cada etapa, que aparecem nos denominadores das expressões (3.12), (3.13) e (3.14), forem bastante menores do que os elementos colocados nos numeradores (aqueles que se pretendem anular). Um multiplicador desta grandeza, ao multiplicar por elementos afectados de erros, aumenta significativamente estes erros de arredondamento e vai propagá-los para as operações aritméticas seguintes. O processo numérico torna-se pois instável.

Este inconveniente ultrapassa-se calculando multiplicadores que sejam, em módulo, menores ou iguais a um. Consegue-se isto, colocando, antes de se iniciar a etapa i, sobre a posição (i, i) (posição 'pivot') da diagonal principal, o elemento da *coluna* i que tem maior módulo - a selecção deve ser feita dentre os elementos que estão *sobre e abaixo da diagonal principal*, isto é, dentre os a_{ji} com $j = i, i+1, \ldots, n$.

A troca (operação elementar 1.) deve ser feita entre as *linhas completas* da matriz ampliada do sistema, para que o sistema resultante seja equivalente.

A esta selecção, seguida da troca entre linhas, chama-se *pivotagem parcial*.

Nesta situação, os denominadores dos multiplicadores são sempre, em módulo, maiores ou iguais aos numeradores e os multiplicadores resultantes são *menores ou iguais a um*, em valor absoluto.

Assim, antes de se iniciar a etapa i, deve-se verificar a necessidade de efectuar trocas de linhas por forma a garantir que sobre a posição da diagonal principal (i,i) fica o elemento de maior módulo.

Os restantes cálculos que levam à condensação da matriz $(A\,|\,b)$ à forma $(U\,|\,c)$ são idênticos aos descritos em 3.4.2.

A fase final de substituição inversa usa as mesmas equações (3.15) e (3.16). No algoritmo 3.3 estão os passos respeitantes a este processo de eliminação de Gauss com pivotagem parcial.

Algoritmo 3.3 :

1. ler n, calcular tol e para $i = 1,\ldots,n$ ler b_i e $(a_{ij}, j = 1,\ldots,n)$
2. para $j = 1,\ldots,n-1$ fazer
   - 2.1. para $k = j,\ldots,n$ calcular piv tal que $|a_{piv\,j}| \geq |a_{kj}|$
   - 2.2. se $piv \neq j$ então trocar b_{piv} com b_j e para $k = j,\ldots,n$ trocar $a_{piv\,k}$ com a_{jk}
   - 2.3. se $|a_{jj}| \leq tol$ então terminar, a matriz A é singular
   - 2.4. para $i = j+1,\ldots,n$ calcular $m_{ij} = -\frac{a_{ij}}{a_{jj}}$
     - 2.4.1. para $k = j+1,\ldots,n$ calcular $a_{ik} = a_{ik} + m_{ij}a_{jk}$
     - 2.4.2. calcular $b_i = b_i + m_{ij}b_j$
3. se $|a_{nn}| \leq tol$ então terminar, a matriz é singular
4. calcular $x_n = \frac{b_n}{a_{nn}}$
5. para $i = n-1,\ldots,1$ calcular $soma = \sum_{j=i+1}^{n} a_{ij}x_j$ e $x_i = \frac{b_i - soma}{a_{ii}}$
6. terminar com $x^* \leftarrow (x_i, i = 1,\ldots,n)$.

3.4.5 Implementação. Rotina GAUPIV

A rotina GAUPIV implementa o algoritmo 3.3. No exemplo seguinte, o sistema é o mesmo do exemplo 3.5.

Exemplo 3.6 Usando pivotagem parcial na resolução do sistema do exemplo 3.5, a condensação envolve duas etapas. Antes de iniciar a primeira etapa, houve necessidade de trocar a linha 1 com a linha 2. Os multiplicadores são então, $m_{21} = -0.333333$ e $m_{31} = -0.666667$. Antes da

segunda etapa trocou-se a linha 2 com a linha 3. O multiplicador passou a ser $m_{32} = 0.1$. A matriz $(U \mid c)$ obtida, com seis casas decimais, é

$$\left(\begin{array}{ccc|c} 9 & 7 & -9 & 1 \\ 0 & 3.333333 & -2 & 0.333333 \\ 0 & 0 & 0.8 & 0.7 \end{array}\right)$$

e a solução (usando as equações (3.15) e (3.16)) é $(0.5, 0.625, 0.875)^T$, igual à calculada por GAUSS. Neste exemplo, os erros de arredondamento cometidos foram insignificantes e a sua propagação ao longo da resolução do exemplo 3.5, pelo algoritmo 3.2, não afectou os resultados finais. O mesmo não se poderá dizer do exemplo 3.2 onde os algoritmos 3.2 e 3.3 fornecem resultados diferentes.

Exemplo 3.7 A resolução do sistema, por um método directo e estável (rotina GAUPIV)

$$\left\{\begin{array}{lcl} 80x_1 + 30x_3 + 10x_4 & = & 40 \\ 80x_2 + 10x_3 + 10x_4 & = & 27 \\ 16x_1 + 20x_2 + 60x_3 + 72x_4 & = & 31 \\ 4x_1 + 8x_4 & = & 2 \end{array}\right.$$

envolve três etapas. Os multiplicadores calculados são $m_{21} = 0$, $m_{31} = -0.2$ e $m_{41} = -0.05$ na primeira etapa, $m_{32} = -0.25$ e $m_{42} = 0$ na segunda etapa e $m_{43} = 0.029126$ na terceira etapa. A solução final é $(0.4, 0.3, 0.25, 0.05)^T$.

3.4.6 Sistemas tridiagonais

Um sistema tridiagonal é aquele em que a matriz dos coeficientes do sistema é tridiagonal. Os elementos não nulos encontram-se sobre a diagonal principal e sobre as diagonais acima e abaixo da diagonal principal. Vamos designar os elementos da diagonal principal por $d_1, d_2, \ldots, d_n$, os elementos da diagonal acima da diagonal principal por $s_1, s_2, \ldots, s_{n-1}$ e os que estão na diagonal abaixo da diagonal principal por $i_2, i_3, \ldots, i_n$. A matriz dos coeficientes tem o seguinte aspecto,

$$\left(\begin{array}{ccccc} d_1 & s_1 & & & \\ i_2 & d_2 & s_2 & & \\ & & \ldots & & \\ & & i_{n-1} & d_{n-1} & s_{n-1} \\ & & & i_n & d_n \end{array}\right)$$

e o vector independente é $b = (b_1, b_2, \ldots, b_n)^T$. Para a resolução directa deste sistema usa-se um processo idêntico ao usado no caso do sistema $Ax = b$. A condensação da matriz dos coeficientes à forma triangular superior torna-se mais simples e rápida uma vez que a maior parte dos elementos da matriz é já igual a zero. Assim, em cada etapa j só temos de anular um elemento, o que se encontra abaixo da posição da diagonal principal, i_{j+1}, $j = 1, \ldots, n-1$. Verifica-se uma redução significativa no número de operações a efectuar. Na etapa j, o correspondente multiplicador é dado por

$$m = -\frac{i_{j+1}}{d_j},$$

e são alterados apenas os seguintes elementos da linha $j+1$,

$$i_{j+1} \leftarrow 0$$

$$d_{j+1} \leftarrow d_{j+1} + m\, s_j$$

e

$$b_{j+1} \leftarrow b_{j+1} + m\, b_j,$$

conservando-se o s_{j+1}, $j = 1, \ldots, n-1$. Após a condensação, o sistema equivalente tem a forma

$$\begin{cases} d_1 x_1 & +s_1 x_2 & & & & = b_1 \\ & d_2 x_2 & +s_2 x_3 & & & = b_2 \\ & & & \cdots & & \\ & & & d_{n-1} x_{n-1} & +s_{n-1} x_n & = b_{n-1} \\ & & & & d_n x_n & = b_n \end{cases}$$

e o passo seguinte consiste em resolver em ordem a x_j, $j = 1, \ldots, n$ por substituição inversa, isto é, da última equação retiramos o valor de x_n,

$$x_n = \frac{b_n}{d_n}$$

e das seguintes retiramos,

$$x_j = \frac{b_j - s_j x_{j+1}}{d_j}, \text{ para } j = n-1, \ldots, 1.$$

O algoritmo correspondente a esta simplificação do método de eliminação de Gauss aplicado a sistemas tridiagonais é o seguinte:

Algoritmo 3.4 :

1. ler n, calcular tol, para $k = 1, \ldots, n$ ler d_k e b_k e para $k = 1, \ldots, n-1$ ler s_k e i_{k+1}
2. para $j = 1, \ldots, n-1$ fazer
   2.1. se $|d_j| \leq tol$ então terminar sem solução
   2.2. calcular $m_j = -\frac{i_{j+1}}{d_j}$, $d_{j+1} = d_{j+1} + m_j s_j$ e $b_{j+1} = b_{j+1} + m_j b_j$
3. se $|d_n| \leq tol$ então terminar sem solução
4. calcular $x_n = \frac{b_n}{d_n}$
5. para $j = n-1, \ldots, 1$ calcular $x_j = \frac{b_j - s_j x_{j+1}}{d_j}$
6. terminar com $x^* \leftarrow (x_j, j = 1, \ldots, n)$.

3.4.7 Implementação. Rotina TRIDIA

A implementação do algoritmo 3.4 origina a rotina TRIDIA.

Exemplo 3.8 Os resultados obtidos na resolução do sistema tridiagonal

$$\begin{cases} 3x_1 & +x_2 & & & = & 5 \\ x_1 & +3x_2 & +x_3 & & = & 10 \\ & x_2 & +3x_3 & +x_4 & = & 15 \\ & & x_3 & +3x_4 & = & 15 \end{cases}$$

são $x_1 = 1$, $x_2 = 2$, $x_3 = 3$ e $x_4 = 4$.

3.4.8 Inversa de uma matriz

Considere uma matriz A quadrada de ordem n.

Teorema 3.3 : Uma matriz quadrada tem inversa (é invertível ou é não singular) se e só se a sua característica é igual à ordem.

A *inversa*, A^{-1}, da matriz A é uma matriz quadrada de ordem n que verifica

$$AA^{-1} = I = A^{-1}A.$$

O cálculo da inversa A^{-1} reduz-se à *resolução de um conjunto de n sistemas de equações lineares* da forma

$$Ax_j = e_j,\ j = 1, \ldots, n,$$

em que as matrizes dos coeficientes dos n sistemas *são constantes e iguais* a A, e os vectores independentes, $e_j \in \mathbf{R}^n$, são as colunas da matriz identidade. Por exemplo, $e_1 = (1, 0, 0, \ldots, 0)^T$ e $e_{n-1} = (0, 0, \ldots, 0, 1, 0)^T$. O vector solução $x_j \in \mathbf{R}^n$ do sistema j, é a coluna j da matriz inversa, $j = 1, \ldots, n$. Como a matriz A é comum, os n sistemas são condensados simultaneamente. A matriz ampliada é $(A \,|\, I)$, em que A é a matriz comum a todos os sistemas e a matriz identidade, I, é composta pelo conjunto dos n vectores independentes do lado direito dos sistemas. Para assegurar estabilidade no processo, a condensação à forma $(U \,|\, J)$, sendo U a matriz triangular superior e J a matriz resultante das transformações elementares efectuadas, deve ser feita pelo processo de *eliminação de Gauss com pivotagem parcial*, tal como foi feita em 3.4.4:

$$(A \,|\, I) \ \rightarrow \ (U \,|\, J).$$

Após a condensação vertical, o passo seguinte consiste na *substituição inversa*, utilizando as equações (3.15) e (3.16). O vector do lado direito, c (de (3.15) e (3.16)), é agora substituído pelas colunas da matriz J.

No entanto, como temos n vectores independentes, que são as colunas da matriz J, e que correspondem aos n sistemas, o processo de substituição inversa deve ser repetido n vezes, sempre com a mesma matriz U, apenas variando o vector independente. Cada um

destes é uma coluna da matriz de transformação J. A solução de cada um destes sistemas forma uma coluna da inversa. Assim, a solução do primeiro sistema

$$\left(\begin{array}{ccccc|c} u_{11} & u_{12} & u_{13} & \dots & u_{1n} & j_{11} \\ & u_{22} & u_{23} & \dots & u_{2n} & j_{21} \\ & & & \dots & & \\ & & & & u_{nn} & j_{n1} \end{array}\right)$$

é o vector $(x_{11}, x_{21}, \dots, x_{n1})^T$ e será a primeira coluna da inversa; a solução do segundo sistema

$$\left(\begin{array}{ccccc|c} u_{11} & u_{12} & u_{13} & \dots & u_{1n} & j_{12} \\ & u_{22} & u_{23} & \dots & u_{2n} & j_{22} \\ & & & \dots & & \\ & & & & u_{nn} & j_{n2} \end{array}\right)$$

é o vector $(x_{12}, x_{22}, \dots, x_{n2})^T$ que irá ocupar a segunda coluna da inversa, e assim sucessivamente, até se chegar ao último sistema

$$\left(\begin{array}{ccccc|c} u_{11} & u_{12} & u_{13} & \dots & u_{1n} & j_{1n} \\ & u_{22} & u_{23} & \dots & u_{2n} & j_{2n} \\ & & & \dots & & \\ & & & & u_{nn} & j_{nn} \end{array}\right),$$

cuja solução é o vector $(x_{1n}, x_{2n}, \dots, x_{nn})^T$ e que ocupará a última coluna da inversa. Tem-se, assim,

$$A^{-1} = \left(\begin{array}{cccc} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{2n} \\ & & \dots & \\ x_{n1} & x_{n2} & \dots & x_{nn} \end{array}\right).$$

O algoritmo correspondente a este processo encontra-se no algoritmo 3.5. Este fornece, não só a inversa de A, mas também as matrizes U e J da condensação, $JA = U$. Basta especificar no parâmetro *selec* o que se pretende.

Algoritmo 3.5 :

1. `ler` n, *selec* `("J e U" ou "inver"), calcular` *tol* `e para` $i = 1, \dots, n$ `ler` $(a_{ij}, j = 1, \dots, n)$
2. `para` $i = 1, \dots, n$ `e para` $j = i, \dots, n$,
   - 2.1. `se` $j = i$ `então fazer` $b_{ii} = 1$
   - 2.2. `senão fazer` $b_{ji} = b_{ij} = 0$
3. `para` $j = 1, \dots, n-1$ `fazer`
   - 3.1. `para` $k = j, \dots, n$ `calcular` *piv* `tal que` $|a_{piv\,j}| \geq |a_{kj}|$

3.2. se $piv \neq j$ então

3.2.1. para $k = j, \ldots, n$ trocar $a_{piv\,k}$ com a_{jk}

3.2.2. para $k = 1, \ldots, n$ trocar $b_{piv\,k}$ com b_{jk}

3.3. se $|a_{jj}| \leq tol$ então terminar, a matriz A é singular

3.4. para $i = j+1, \ldots, n$ calcular $m_{ij} = -\frac{a_{ij}}{a_{jj}}$

3.4.1. para $k = j+1, \ldots, n$ calcular $a_{ik} = a_{ik} + m_{ij}a_{jk}$

3.4.2. para $k = 1, \ldots, n$ calcular $b_{ik} = b_{ik} + m_{ij}b_{jk}$

4. se $selec$ = "J e U"então terminar com $J \leftarrow ((b_{ij}, j = 1, \ldots, n), i = 1, \ldots, n)$ e $U \leftarrow ((a_{ij}, j = i, \ldots, n), i = 1, \ldots, n)$.

5. para $k = 1, \ldots, n$

5.1. se $|a_{nn}| \leq tol$ então terminar, a matriz A é singular

5.2. calcular $x_{nk} = \frac{b_{nk}}{a_{nn}}$

5.3. para $i = n-1, \ldots, 1$ calcular $soma = \sum_{j=i+1}^{n} a_{ij}x_{jk}$ e $x_{ik} = \frac{b_{ik} - soma}{a_{ii}}$

6. terminar com $A^{-1} \leftarrow ((x_{ij}, j = 1, \ldots, n),\ i = 1, \ldots, n)$.

Se A_c^{-1} for a inversa calculada de A, a matriz *resíduo*, definida por

$$R = AA_c^{-1} - I \tag{3.17}$$

fornece informação relativa à precisão do resultado obtido, isto é, relativa à acumulação de erros de arredondamento em A_c^{-1}. Uma matriz R com elementos próximos de zero, garante uma inversa calculada A_c^{-1} com grande precisão. De facto, multiplicando (3.17) à esquerda por A^{-1} (exacta) fica-se com

$$A^{-1}R = A^{-1}AA_c^{-1} - A^{-1}I$$

ou

$$A_c^{-1} - A^{-1} = A^{-1}R$$

e

$$\delta A^{-1} = A^{-1}R.$$

Em termos de normas, tem-se

$$\|\delta A^{-1}\| \leq \|A^{-1}\|\|R\| \quad \text{e} \quad \frac{\|\delta A^{-1}\|}{\|A^{-1}\|} \leq \|R\|$$

para limite superior do erro relativo no cálculo da inversa.

3.4.9 Implementação. Rotina INVERS

A rotina INVERS implementa o algoritmo 3.5.

Exemplo 3.9 No cálculo da inversa de

$$A = \begin{pmatrix} 1 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{pmatrix},$$

tem-se para a matriz ampliada

$$(A \mid I) = \left(\begin{array}{cccc|cccc} 1 & -1 & 0 & 0 & 1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 2 & 0 & 0 & 0 & 1 \end{array}\right),$$

que após a condensação tem a forma:

$$(U \mid J) = \left(\begin{array}{cccc|cccc} 1 & -1 & 0 & 0 & 1 & 0 & 0 & 0 \\ & 1 & -1 & 0 & 1 & 1 & 0 & 0 \\ & & 1 & -1 & 1 & 1 & 1 & 0 \\ & & & 1 & 1 & 1 & 1 & 1 \end{array}\right).$$

Da substituição inversa, resulta

$$Ux_1 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \Rightarrow x_1 = \begin{pmatrix} 4 \\ 3 \\ 2 \\ 1 \end{pmatrix}; \quad Ux_2 = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 1 \end{pmatrix} \Rightarrow x_2 = \begin{pmatrix} 3 \\ 3 \\ 2 \\ 1 \end{pmatrix};$$

$$Ux_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \end{pmatrix} \Rightarrow x_3 = \begin{pmatrix} 2 \\ 2 \\ 2 \\ 1 \end{pmatrix} \text{ e } Ux_4 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \Rightarrow x_4 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}.$$

Assim

$$A^{-1} = \begin{pmatrix} 4 & 3 & 2 & 1 \\ 3 & 3 & 2 & 1 \\ 2 & 2 & 2 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}.$$

3.5 Decomposição de matrizes. Determinantes

3.5.1 Decomposição triangular L U

A condensação de $Ax = b$, com $A \in \mathbf{R}^{n\times n}$, x e $b \in \mathbf{R}^n$, à forma $Ux = c$ é equivalente à multiplicação à esquerda, de $Ax = b$, por uma matriz J,

$$JAx = Jb, \text{ com } JA = U \text{ e } Jb = c.$$

A matriz J é, em geral, uma matriz triangular inferior e obtém-se do produto de várias matrizes triangulares inferiores que surgem durante as $n-1$ etapas do processo de condensação. Se ao longo do processo não houve necessidade de trocar linhas para garantir estabilidade,

$$J = J_{n-1} J_{n-2} \ldots J_2 J_1, \tag{3.18}$$

e cada J_i corresponde a um conjunto de *operações elementares* executadas sobre A do tipo: *substituição de uma linha pela que dela se obtém adicionando o produto de outra linha por um escalar*. Por exemplo, J_1 deriva da matriz identidade substituindo os zeros das posições $(2,1),(3,1),\ldots$ e $(n,1)$, respectivamente, pelos multiplicadores da etapa 1 da condensação, $m_{21}, m_{31}, \ldots$ e m_{n1},

$$\begin{pmatrix} 1 & 0 & 0 & \ldots & 0 \\ m_{21} & 1 & 0 & \ldots & 0 \\ m_{31} & 0 & 1 & \ldots & 0 \\ & & & \ldots & \\ m_{n1} & 0 & 0 & \ldots & 1 \end{pmatrix}.$$

A matriz J_2 deriva da identidade substituindo os zeros das posições $(3,2)$, $(4,2),\ldots$ e $(n,2)$ respectivamente pelos multiplicadores da segunda etapa, $m_{32}, m_{42}, \ldots$ e m_{n2} e assim sucessivamente. A matriz J_{n-1} obtém-se da I substituindo o zero da posição $(n, n-1)$ por $m_{n\,n-1}$, que é o multiplicador da última etapa, de ordem $n-1$.

Por sua vez, se houve necessidade de trocar linhas, a matriz J terá o seguinte aspecto

$$J = J_{n-1}\, I_{n-1,\,n}\, J_{n-2}\, I_{n-2,\,k'}\, \ldots J_2\, I_{2,\,k''}\, J_1\, I_{1,\,k'''} \tag{3.19}$$

em que as matrizes J_i, $i = 1, 2, \ldots, n-1$ são idênticas às do caso anterior e as $I_{i,j}$, que *podem ou não existir em todas as etapas*, são matrizes que executam sobre A (quando multiplicadas à esquerda de A) operações elementares do tipo: *troca das duas linhas paralelas i e j*. Os índices k', k'' e k''' podem tomar quaisquer dos seguintes valores: $k' = n-1$ ou n, $k'' = 3, 4, \ldots$ ou n e $k''' = 2, 3, \ldots$ ou n. Por exemplo, o aspecto da matriz $I_{1,4}$ (que vem da matriz identidade de ordem 5 com as linhas 1 e 4 trocadas), num problema de ordem 5, é

$$\begin{pmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix},$$

e que multiplicando, à esquerda de A, troca as linhas 1 e 4 de A.

Se a matriz J for não singular, J^{-1} existe e tem-se

$$A = J^{-1}U \quad \text{bem como} \quad b = J^{-1}c.$$

Designando J^{-1} por L, a decomposição de A em LU é conhecida por *decomposição triangular* e as matrizes L e U são os *factores triangulares* de A. A matriz L constrói-se

etapa a etapa, durante o processo de condensação de A em U, uma vez que

$$L = J^{-1} = I_{1,k'''}\, J_1^{-1}\, I_{2,k''}\, J_2^{-1} \ldots I_{n-2,k'}\, J_{n-2}^{-1}\, I_{n-1,n}\, J_{n-1}^{-1}.$$

A inversa de cada matriz J_i, $(i = 1, \ldots, n-1)$ obtém-se da J_i trocando os sinais aos elementos multiplicadores que a compõem.

A matriz U da decomposição de A em LU é sempre uma matriz triangular superior.

As matrizes J_i de (3.18) são triangulares inferiores e têm uns sobre a diagonal principal.

Quando não se efectuam trocas de linhas, não existem as matrizes $I_{i,j}$, donde se conclui que J também é triangular inferior e tem uns sobre a diagonal principal. A inversa L de uma matriz triangular inferior J é também uma matriz triangular inferior e neste caso conserva os uns sobre a diagonal principal. Razão porque esta decomposição é conhecida por decomposição triangular.

No entanto, se durante a condensação de A à forma U houve necessidade de trocar linhas, a matriz resultante J com a forma apresentada em (3.19), *não tem* a forma de uma matriz triangular inferior, embora as J_i, $i = 1, \ldots, n-1$ o sejam. As matrizes $I_{i,j}$, que aparecem na definição de J, trocam as linhas i e j à matriz que se encontrar à sua direita. A forma triangular inferior de J pode ser reconstruída se se efectuarem, sobre J, as trocas de linhas feitas anteriormente durante a condensação. O algoritmo 3.6 calcula os factores triangulares LU de uma matriz A e fornece, através do parâmetro s, o número de vezes que se efectuaram trocas de linhas.

Algoritmo 3.6 :

1. ler n, calcular tol, para $i = 1, \ldots, n$ ler $(a_{ij}, j = 1, \ldots, n)$ e fazer $s = 0$
2. para $j = 1, \ldots, n-1$ fazer
   - 2.1. para $k = j, \ldots, n$ calcular piv tal que $|a_{piv\,j}| \geq |a_{kj}|$
   - 2.2. se $piv \neq j$ então fazer $s = s + 1$ e para $k = j, \ldots, n$ trocar $a_{piv\,k}$ com a_{jk}
   - 2.3. se $|a_{jj}| \leq tol$ então terminar, a matriz A é singular
   - 2.4. para $i = 1, \ldots, n$ e $k = 1, \ldots, n$
     - 2.4.1. se $k = i$ então fazer $m_{ii} = 1$
     - 2.4.2. senão fazer $m_{ki} = m_{ik} = 0$
   - 2.5. para $i = j+1, \ldots, n$ calcular $m_{ij} = -\frac{a_{ij}}{a_{jj}}$
     - 2.5.1. para $k = j+1, \ldots, n$ calcular $a_{ik} = a_{ik} + m_{ij} a_{jk}$
     - 2.5.2. fazer $m_{ij} = -m_{ij}$
   - 2.6. se $j = 1$ então
     - 2.6.1. se $piv \neq j$ então para $k = 1, \ldots, n$ trocar (linhas) $m_{piv\,k}$ com m_{jk}
     - 2.6.2. para $i = 1, \ldots, n$ e $k = 1, \ldots, n$ fazer $prod_{ik} = m_{ik}$
   - 2.7. senão
     - 2.7.1. se $piv \neq j$ então para $k = 1, \ldots, n$ trocar (colunas) $prod_{k\,piv}$ com $prod_{kj}$
     - 2.7.2. para $i = 1, \ldots, n$ e $k = 1, \ldots, n$ calcular $aux_{ik} = \sum_{l=1}^{n} prod_{il}\, m_{lk}$

2.7.3. para $i = 1, \ldots, n$ e $k = 1, \ldots, n$ fazer $prod_{ik} = aux_{ik}$

3. terminar com $L \leftarrow ((prod_{ij}, j = 1, \ldots, n), i = 1, \ldots, n)$, $U \leftarrow ((a_{ij}, j = i, \ldots, n), i = 1, \ldots, n)$ e $n^{o.}$ de trocas de linhas $\leftarrow s$.

3.5.2 Implementação. Rotina TRIANG

A implementação do algoritmo 3.6 origina a rotina TRIANG.

Exemplo 3.10 O cálculo dos factores triangulares de

$$A = \begin{pmatrix} 2 & 1 & 0 & 0 \\ 4 & 5 & 2 & 0 \\ 0 & -6 & 0 & 3 \\ 0 & 0 & -4 & -2 \end{pmatrix}$$

pela rotina TRIANG origina

$$L = \begin{pmatrix} 0.5 & 0.25 & 0.25 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \quad U = \begin{pmatrix} 4 & 5 & 2 & 0 \\ 0 & -6 & 0 & 3 \\ 0 & 0 & -4 & -2 \\ 0 & 0 & 0 & -0.25 \end{pmatrix} \quad \text{com} \quad s = 3.$$

A matriz L obtém-se pelo seguinte processo (a partir de (3.19)):

$$L = (J)^{-1} = (J_3\, I_{3,4}\, J_2\, I_{2,3}\, J_1\, I_{1,2})^{-1},$$

com

$$J_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -0.5 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \text{ e } I_{1,2} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix};$$

$$J_2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -0.25 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \text{ e } I_{2,3} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix};$$

$$J_3 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -0.25 & 1 \end{pmatrix} \text{ e } I_{3,4} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

Multiplicando a matriz L, à esquerda, por $I_{3,4}\, I_{2,3}\, I_{1,2}$ (que corresponde a ter trocado as linhas 1 e 2, na primeira etapa da condensação, as linhas 2 e 3, na segunda etapa, e as linhas 3 e 4, na terceira etapa) obtém-se L, com a forma triangular superior com *uns* sobre a diagonal.

Exemplo 3.11 O cálculo dos factores triangulares de

$$A = \begin{pmatrix} 5 & 2 & 1 \\ 5 & -6 & 2 \\ -4 & 2 & 1 \end{pmatrix}$$

pela rotina TRIANG origina, com $s = 0$,

$$L = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ -0.8 & -0.45 & 1 \end{pmatrix} \text{ e } U = \begin{pmatrix} 5 & 2 & 1 \\ 0 & -8 & 1 \\ 0 & 0 & 2.25 \end{pmatrix}.$$

Neste caso, a matriz L é triangular inferior com uns sobre a diagonal principal, como se esperava.

3.5.3 Factorização Cholesky

Definição 3.4 : Se os determinantes das submatrizes principais, A_i, de ordem i, para $i = 1, \ldots, n$, de uma matriz A, quadrada de ordem n, são positivos, então A diz-se definida positiva.

Para uma matriz A definida positiva, tem-se

$$x^T Ax > 0, \text{ para qualquer vector } x \neq 0.$$

Teorema 3.4 : Se A é uma matriz real do tipo $n \times n$, simétrica e definida positiva, então existe uma matriz triangular inferior L, real e do tipo $n \times n$, tal que $A = LL^T$.

A factorização de uma matriz A simétrica e definida positiva na forma $A = LL^T$ é conhecida por *factorização Cholesky*[4].

Se os elementos da matriz A forem designados por a_{ij}, $(i, j = 1, \ldots, n)$ e a matriz L, triangular inferior, definida por

$$L = \begin{pmatrix} l_{11} & & & \ldots & \\ l_{21} & l_{22} & & \ldots & \\ & & & \ldots & \\ l_{n1} & l_{n2} & l_{n3} & \ldots & l_{nn} \end{pmatrix}$$

então, a partir de $A = LL^T$, é possível estabecer relações entre os seus elementos. Assim,

$$l_{ii} = \sqrt{a_{ii} - \sum_{k=1}^{i-1} l_{ik}^2} \text{ para } i = 1, \ldots, n \tag{3.20}$$

e

$$l_{ji} = \frac{1}{l_{ii}}(a_{ji} - \sum_{k=1}^{i-1} l_{jk} l_{ik}) \text{ para } j = i+1, \ldots, n. \tag{3.21}$$

[4] A. L. Cholesky foi um major francês que viveu entre 1875 e 1918. Entre 1906 a 1909 esteve na ilha de Creta e mais tarde no Norte de África onde desenvolveu trabalhos de investigação no domínio da Geodesia. Desenvolveu um método para calcular a solução de um problema de ajuste de dados do tipo dos mínimos quadrados e a factorização Cholesky de uma matriz, embora com o seu nome, pode também ser deduzida a partir de um teorema, bem mais antigo, devido a Jacobi (1804 - 1851) (Hämmerlin e Hoffmann (1991)).

Note-se que $\sum_{k=1}^{i} l_{ik}^2 = a_{ii}, i = 1, \ldots, n$.

Se, na resolução de um sistema $Ax = b$, a matriz dos coeficientes A é simétrica e definida positiva, o cálculo da matriz L, da sua factorização Cholesky, origina uma resolução muito simplificada de $Ax = b$. Como $A = LL^T$, $Ax = b$ reduz-se a $LL^T x = b$ ou seja

$$Lc = b, \quad \text{por substituição directa}$$

seguida de

$$L^T x = c, \quad \text{por substituição inversa.}$$

O algoritmo 3.7 apresenta os passos relativos à factorização Cholesky de A, bem como à resolução do sistema baseado numa matriz A simétrica e definida positiva. Basta especificar no parâmetro *selec* o que se pretende.

Algoritmo 3.7 :

1. ler n, *selec* ("factor L" ou "sistema"), calcular *tol* e para $i = 1, \ldots, n$
   ler $(a_{ji}, j = i, \ldots, n)$ (matriz simétrica)
2. para $i = 1, \ldots, n$ fazer
   2.1. se $i = 1$ então
      2.1.1. se $a_{ii} \leq tol$ então terminar, a matriz não é definida positiva
      2.1.2. calcular $l_{ii} = \sqrt{a_{ii}}$
   2.2. senão
      2.2.1. calcular $soma = \sum_{k=1}^{i-1} l_{ik}^2$ e $r = a_{ii} - soma$
      2.2.2. se $r \leq tol$ então terminar, a matriz não é definida positiva
      2.2.3. calcular $l_{ii} = \sqrt{r}$
   2.3. para $j = i + 1, \ldots, n$
      2.3.1. se $i = 1$ então calcular $l_{ji} = \frac{a_{ji}}{l_{ii}}$
      2.3.2. senão calcular $soma = \sum_{k=1}^{i-1} l_{jk} l_{ik}$ e $l_{ji} = \frac{a_{ji} - soma}{l_{ii}}$
3. se *selec* = "factor L"então terminar com $L \leftarrow ((l_{ji}, j = i, \ldots, n), i = 1, \ldots, n)$
4. para $i = 1, \ldots, n$ ler b_i
5. calcular $c_1 = \frac{b_1}{l_{11}}$
6. para $i = 2, \ldots, n$ calcular $soma = \sum_{j=1}^{i-1} l_{ij} c_j$ e $c_i = \frac{b_i - soma}{l_{ii}}$
7. calcular $x_n = \frac{c_n}{l_{nn}}$
8. para $i = n - 1, \ldots, 1$ calcular $soma = \sum_{j=i+1}^{n} l_{ji} x_j$ e $x_i = \frac{c_i - soma}{l_{ii}}$
9. terminar com $x^* \leftarrow (x_i, i = 1, \ldots, n)$.

3.5.4 Implementação. Rotina CHOLES

O algoritmo 3.7 foi implementado tendo originado a rotina CHOLES.

Exemplo 3.12 No cálculo da decomposição Cholesky de

$$A = \begin{pmatrix} 10 & 1 & 4 & 0 \\ 1 & 10 & 5 & -1 \\ 4 & 5 & 10 & 7 \\ 0 & -1 & 7 & 9 \end{pmatrix} \text{ encontra-se } L = \begin{pmatrix} 3.162278 & 0 & 0 & 0 \\ 0.316228 & 3.146427 & 0 & 0 \\ 1.264911 & 1.461976 & 2.502524 & 0 \\ 0 & -0.317821 & 2.982847 & 0.040161 \end{pmatrix}.$$

3.5.5 Factorização Q R

Definição 3.5 : Uma matriz do tipo $n \times n$ da forma

$$Q = I - 2ww^T, \quad com \quad w^T w = 1$$

em que w é um vector não nulo de $\mathbb{R}^n$, chama-se matriz Householder.

Definição 3.6 : Uma matriz Q diz-se ortogonal se é real e se

$$QQ^T = Q^T Q = I.$$

Teorema 3.5 : Dada uma matriz real A, quadrada de ordem n, existe uma matriz ortogonal Q e uma triangular superior R, tal que $A = QR$. Se a matriz A é não singular, a correspondente decomposição é única. É possível assegurar que todos os elementos da diagonal de R sejam não negativos.

A decomposição de A em QR chama-se *decomposição Q R* e é baseada nas matrizes de transformação Householder[5], de cada uma das etapas, que são ortogonais e simétricas, $Q_i^{-1} = Q_i^T = Q_i$. Na resolução de um sistema de equações $Ax = b$, a decomposição QR de A origina as seguintes etapas:

1. decomposição $Q_{n-1} \ldots Q_2 Q_1 A = R$ ou, $A = Q_1 Q_2 \ldots Q_{n-1} R$,

[5]A. S. Householder nasceu em 1904. Em 1925 concluíu o B.A. na Universidade de Northwestern, em 1927 o M.A. na Universidade de Cornell e, em 1937, o Ph.D. em Matemática, na Universidade de Chicago dos Estados Unidos da América. Embora Householder seja mais conhecido pela sua contribuição na Álgebra Linear Numérica, nomeadamente com as matrizes hermitianas conhecidas nas transformações de Householder e na utilização das normas para a localização de valores próprios, no início da sua carreira interessou-se pelo Cálculo das Variações e pela Biomatemática. Em 1946 foi trabalhar para a Divisão de Matemática do Laboratório Nacional de Oak Ridge (ORNL) , tendo-se tornado Director em 1948. Foi nesta Divisão que começou a trabalhar em Análise Numérica. Em 1969 deixou o ORNL e tornou-se professor de matemática na Universidade de Tennessee. Dos livros que publicou, os mais conhecidos foram 'Principles of Numerical Analysis' considerado por J. Wilkinson como o primeiro tratamento moderno da Análise Numérica, e 'The Theory of Matrices in Numerical Analysis'. Foi Presidente da Sociedade SIAM e da 'Association for Computing Machinery' e Vice-presidente da 'American Mathematical Society'. Em 1974 reformou-se e foi viver para a California onde morreu em 1993 (SIAM NEWS, Outubro de 1993).

2. como $Ax = b$, também $Q_{n-1} \dots Q_1 Ax = Q_{n-1} \dots Q_1 b$ ou $Rx = Q_1 \dots Q_{n-1} b$, e a resolução é feita por substituição inversa uma vez que R é triangular superior e o produto $Q_{n-1} \dots Q_1 b$ é feito directamente durante a decomposição.

A decomposição envolve $n-1$ etapas. Se Q_1 for a matriz da primeira etapa, Q_2 a matriz da segunda etapa, ..., Q_{n-1} a matriz da última etapa, então $Q_{n-1} \dots Q_2 Q_1 A = R$. Como o objectivo é reduzir A à forma triangular superior, R, na primeira etapa anulam-se os elementos de A, que se encontram nas posições $(2,1), (3,1), \dots, (n,1)$, usando o vector

$$w = (w_1, w_2, \dots, w_n)^T \text{ com } w_1^2 + w_2^2 + \dots + w_n^2 = 1$$

definindo para isso

$$w_1^2 = \frac{1}{2}(1 \pm \frac{a_{11}}{\sqrt{a_{11}^2 + a_{21}^2 + \dots + a_{n1}^2}}),$$

$$w_2 = \pm \frac{a_{21}}{2w_1\sqrt{a_{11}^2 + a_{21}^2 + \dots + a_{n1}^2}},$$

$$\dots$$

$$w_n = \pm \frac{a_{n1}}{2w_1\sqrt{a_{11}^2 + a_{21}^2 + \dots + a_{n1}^2}}.$$

A matriz Q_1, de acordo com a definição 3.5 tem o seguinte aspecto,

$$\begin{pmatrix} 1-2w_1^2 & -2w_1w_2 & \dots & -2w_1w_n \\ -2w_2w_1 & 1-2w_2^2 & \dots & -2w_2w_n \\ & & \dots & \\ -2w_nw_1 & -2w_nw_2 & \dots & 1-2w_n^2 \end{pmatrix}$$

e no fim da primeira etapa tem-se, considerando $A_1 = A$,

$$A_2 = Q_1 A_1 = \begin{pmatrix} \times & \times & \dots & \times \\ 0 & \times & \dots & \times \\ & & \dots & \\ 0 & \times & \dots & \times \end{pmatrix}.$$

O sinal $\times$ indica que o elemento foi transformado mas é diferente de zero. A segunda etapa anula os elementos de A_2 que se encontram nas posições $(3,2), (4,2), \dots (n,2)$. O vector w é agora definido por

$$(0, w_2, w_3, \dots, w_n)^T \text{ com } w_2^2 + \dots + w_n^2 = 1$$

sendo

$$w_2^2 = \frac{1}{2}(1 \pm \frac{a_{22}}{\sqrt{a_{22}^2 + a_{32}^2 + \dots + a_{n2}^2}}),$$

$$w_3 = \pm \frac{a_{32}}{2w_2\sqrt{a_{22}^2 + a_{32}^2 + \cdots + a_{n2}^2}},$$
$$\cdots$$
$$w_n = \pm \frac{a_{n2}}{2w_2\sqrt{a_{22}^2 + a_{32}^2 + \cdots + a_{n2}^2}},$$

em função dos elementos a_{ij} de A_2. A matriz Q_2 é definida por

$$\begin{pmatrix} 1 & 0 & 0 & \ldots & 0 \\ 0 & 1-2w_2^2 & -2w_2w_3 & \ldots & -2w_2w_n \\ & & & \ldots & \\ 0 & -2w_nw_2 & -2w_nw_3 & \ldots & 1-2w_n^2 \end{pmatrix}$$

e a matriz A_3 tem a forma:

$$A_3 = Q_2A_2 = \begin{pmatrix} \times & \times & \times & \ldots & \times \\ 0 & \times & \times & \ldots & \times \\ 0 & 0 & \times & \ldots & \times \\ & & & \ldots & \\ 0 & 0 & \times & \ldots & \times \end{pmatrix}.$$

Este processo é repetido até se chegar à última etapa, de ordem $n-1$, em que $w = (0, 0, \ldots, 0, w_{n-1}, w_n)^T$, $w_{n-1}^2 + w_n^2 = 1$ e

$$w_{n-1}^2 = \frac{1}{2}(1 \pm \frac{a_{n-1\,n-1}}{\sqrt{a_{n-1\,n-1}^2 + a_{n\,n-1}^2}}),$$

$$w_n = \pm \frac{a_{n\,n-1}}{2w_n\sqrt{a_{n-1\,n-1}^2 + a_{n\,n-1}^2}},$$

em função dos elementos a_{ij} da matriz A_{n-1}. A matriz transformada é $A_n = Q_{n-1}A_{n-1} = R$. Os passos do algoritmo que correspondem a esta decomposição são apresentados no algoritmo 3.8. Além das matrizes $Q = Q_1Q_2 \ldots Q_{n-1}$ e R, de $A = QR$, o algoritmo fornece também o parâmetro s, com o número de matrizes de decomposição, Q_i, que são diferentes da identidade.

Algoritmo 3.8 :

1. `ler` n, `calcular` tol, `para` $i = 1, \ldots, n$ `ler` $(a_{ij}, j = 1, \ldots, n)$ `e fazer` $s = n - 1$
2. `para` $i = 1, \ldots, n$ `e` $j = i, \ldots, n$
   - 2.1. `se` $j = i$ `então fazer` $aux_{ij} = 1$
   - 2.2. `senão fazer` $aux_{ij} = aux_{ji} = 0$
3. `para` $k = 1, \ldots, n-1$
   - 3.1. `calcular` $den = \sqrt{\sum_{j=k}^{n} a_{jk}^2}$

3.2. se $den \leq tol$ então incrementar o índice k e regressar ao passo 3.1

3.3. para $i = k, \ldots, n$ fazer

3.3.1. se $i = k$ então calcular $w_i = \sqrt{\frac{1}{2}(1 + sinal(a_{kk})\frac{a_{kk}}{den})}$

3.3.2. senão calcular $w_i = sinal(a_{kk})\frac{a_{ik}}{2w_k den}$

3.4. para $i = 1, \ldots, n$

3.4.1. se $i < k$ então fazer $q_{ii} = 1$

3.4.2. senão fazer $q_{ii} = 1 - 2w_i^2$

3.4.3. para $j = i + 1, \ldots, n$

3.4.3.1. se $i < k$ então fazer $q_{ij} = q_{ji} = 0$

3.4.3.2. senão fazer $q_{ij} = q_{ji} = -2w_i w_j$

3.5. para $i = 1, \ldots, n$

3.5.1. se $q_{ii} \neq 1$ então fazer identidade=.FALSE. e ir para o passo 3.7

3.5.2. para $j = i + 1, \ldots, n$, se $q_{ji} \neq 0$ então fazer identidade=.FALSE. e ir para o passo 3.7

3.6. fazer identidade=.TRUE. e ir para o passo 3.10

3.7. para $i = 1, \ldots, n$ e $j = 1, \ldots, n$ calcular $prod_{ij} = \sum_{l=1}^{n} q_{il} a_{lj}$

3.8. para $i = 1, \ldots, n$ e $j = 1, \ldots, n$ calcular $ort_{ij} = \sum_{l=1}^{n} aux_{il} q_{lj}$

3.9. para $i = 1, \ldots, n$ e $j = 1, \ldots, n$ fazer $a_{ij} = prod_{ij}$ e $aux_{ij} = ort_{ij}$

3.10. se identidade=.TRUE. então fazer $s = s - 1$

4. terminar com $Q \leftarrow ((ort_{ij}, j = 1, \ldots, n), i = 1, \ldots, n)$, $R \leftarrow ((a_{ij}, j = i, \ldots, n), i = 1, \ldots, n)$ e $n^{o.}$ de matrizes $Q(\neq I) \leftarrow s$.

Uma vantagem do método de Householder, na resolução de um sistema de equações lineares, quando comparado com o método de eliminação de Gauss, do ponto de vista da estabilidade numérica, está relacionada com o número de condição da matriz do sistema $Rx = Q_{n-1} \ldots Q_1 b$, que é igual ao número de condição da matriz do sistema original $Ax = b$. Não há necessidade de trocar linhas para que se conserve a estabilidade. Um maior número de operações efectuadas é o seu grande inconveniente.

3.5.6 Implementação. Rotina QSUPER

A rotina QSUPER resulta da implementação do algoritmo 3.8.

Exemplo 3.13 Os factores Q e R obtidos para a matriz

$$A = \begin{pmatrix} 5 & 2 & 1 \\ 5 & -6 & 2 \\ -4 & 2 & 1 \end{pmatrix},$$

são $Q = Q_1 Q_2 =$

$$\begin{pmatrix} -0.615457 & 0.727158 & 0.304061 \\ -0.615457 & -0.684384 & 0.390935 \\ 0.492366 & 0.053468 & 0.868744 \end{pmatrix}, \quad R = \begin{pmatrix} -8.124038 & 3.446562 & -1.354006 \\ 0 & 5.667558 & -0.588143 \\ 0 & 0 & 1.954675 \end{pmatrix}$$

e $s = 2$.

3.5.7 Determinante de uma matriz

O cálculo do *determinante* de uma matriz A, quadrada de ordem n, deve ser feito tendo como base a decomposição da matriz A. O processo torna-se então mais simples face à utilização das seguintes propriedades dos determinantes:

Propriedade 3.1 : O determinante de um produto de matrizes da mesma ordem é igual ao produto dos determinantes das matrizes factores.

Propriedade 3.2 : Se a matriz A é triangular então

$$det\,(A) = \prod_{i=1}^{n} a_{ii}.$$

Em termos de estabilidade numérica, a decomposição triangular de A em LU deve ser feita pelo método de eliminação de Gauss com pivotagem parcial. As diversas matrizes J_i do processo de decomposição de A em LU, são triangulares inferiores (veja-se em 3.5.1, caso da equação (3.18)) com *uns* sobre a diagonal e os respectivos determinantes são sempre iguais a um. As inversas das matrizes J_i são do mesmo tipo das J_i, portanto têm determinantes iguais a um. Se numa etapa houve necessidade de trocar duas linhas, quer dizer que se usou uma matriz do tipo $I_{i,j}$, como o modelo representado em (3.19) indica, e o respectivo determinante é multiplicado por -1. *Cada utilização deste tipo de matriz equivale a multiplicar o determinante por -1.* Assim, o determinante de L ou é igual a um ou a -1. Se s for o número de vezes em que houve necessidade de trocar linhas, então

$$det\,(L) = (-1)^s 1.$$

Como a matriz U é triangular superior, $det(U) = \prod_{i=1}^{n} u_{ii}$ (propriedade 3.2) e

$$det\,(A) = det\,(L)\,det\,(U) = (-1)^s \prod_{i=1}^{n} u_{ii}. \tag{3.22}$$

Se a matriz A é simétrica e definida positiva, a decomposição triangular (factorização Cholesky) resume-se a $A = LL^T$ (veja-se em 3.5.3) e

$$det\,(A) = det\,(L)\,det\,(L^T) = det\,(L)^2 = \prod_{i=1}^{n} l_{ii}^2. \tag{3.23}$$

Se na decomposição de A foram usadas as matrizes Householder, como o determinante de cada matriz Householder Q_i é igual a -1, desde que seja diferente da matriz identidade (que neste caso teria determinante igual a um), o determinante de $A = QR$ é dado por

$$det\,(A) = (-1)^s det\,(R) = (-1)^s \prod_{i=1}^{n} r_{ii}, \tag{3.24}$$

em que os r_{ii} são os elementos da diagonal principal de R, triangular superior (veja-se em 3.5.5), e $s = n-1-ni$, sendo ni o número de matrizes Householder Q_i iguais à identidade.

O algoritmo 3.9 calcula o determinante de uma matriz A, usando uma das três decomposições descritas. A selecção é feita pelo parâmetro *tipo* do algoritmo.

Algoritmo 3.9 :

1. ler n, *tipo* ("L U ", "Q R "ou "L L^T") e para $i = 1, \ldots, n$ ler $(a_{ij}, j = 1, \ldots, n)$
2. escolher a decomposição :
   - 2.1. se *tipo* = "L U "então calcular decomposição L U (algoritmo 3.6), registar s e para $i = 1, \ldots, n$ fazer $u_{ii} = a_{ii}$
   - 2.2. senão
     - 2.2.1. se *tipo* = "Q R "então calcular factorização Q R (algoritmo 3.8), registar s e para $i = 1, \ldots, n$ fazer $u_{ii} = a_{ii}$
     - 2.2.2. senão calcular factorização Cholesky (algoritmo 3.7) e para $i = 1, \ldots, n$ fazer $u_{ii} = l_{ii}$
3. calcular $det = \prod_{i=1}^{n} u_{ii}$
4. se *tipo* = "L L^T"então $det = det^2$ senão se $s = n^{o.}$ impar então fazer $det = -det$
5. terminar com $det(A) \leftarrow det$.

3.5.8 Implementação. Rotina DETERM

A rotina DETERM implementa o algoritmo 3.9. Nos exemplos seguintes é calculado o $det\,(A)$,

Exemplo 3.14 O determinante da matriz do exemplo 3.12, que é simétrica e definida positiva, é igual a $(3.162278 \times 3.146427 \times 2.502524 \times 0.040161)^2 = 1.000002$ arredondando para seis casas decimais. Foi usada a factorização Cholesky para a decomposição de A e a fórmula (3.23) para o cálculo do determinante.

Exemplo 3.15 O determinante de

$$A = \begin{pmatrix} 2 & 1 & 0 & 0 \\ 4 & 5 & 2 & 0 \\ 0 & -6 & 0 & 3 \\ 0 & 0 & -4 & -2 \end{pmatrix}$$

dá igual a 24 pela decomposição LU. Da expressão (3.22) tem-se

$$det\,(A) = (-1)^3(4 \times -6 \times -4 \times -0.25).$$

Relembrar a matriz U e as trocas de linhas efectuadas na condensação da matriz A, do exemplo 3.10.

3.6 Métodos iterativos

3.6.1 Introdução teórica. Convergência

Os *métodos iterativos* para a resolução de um sistema de equações lineares diferem dos métodos directos no que se refere ao número de operações aritméticas a efectuar. Nos métodos iterativos a solução exacta só seria atingida ao fim de uma sequência infinita de operações. No entanto, o processo iterativo termina quando se atinge uma aproximação à solução com uma certa precisão.

Os métodos iterativos, para a resolução de $Ax = b$, assumem uma importância especial quando a matriz dos coeficientes A é de grande dimensão e esparsa, devido à redução que se verifica no número de operações. Estes métodos baseiam-se na partição da matriz A em $M - N$, sendo a matriz M não singular. Na generalidade, o processo é definido pela seguinte equação iterativa (de $Ax = b$)

$$Mx^{(k+1)} = Nx^{(k)} + b, \text{ para } k = 1, 2, \ldots \tag{3.25}$$

Se as matrizes M e N não dependem do índice da iteração, k, o método diz-se *estacionário*. São exemplos de métodos iterativos estacionários os métodos de Jacobi (em 3.6.2) e de Gauss-Seidel (em 3.6.4). Os métodos em que M e N variam com k dizem-se *não estacionários*.

Uma das questões mais importantes directamente relacionada com os métodos iterativos é a da convergência. Só faz sentido implementar um algoritmo iterativo se ele convergir para a solução procurada. Vamos analisar o problema da convergência dos métodos iterativos baseados na equação (3.25). Defina-se o vector $\epsilon^{(k)} = x - x^{(k)}$ como sendo o erro da aproximação calculada na iteração k, em relação à solução exacta, x. Da diferença entre $Mx = Nx + b$ (a solução exacta verifica a equação iterativa) e (3.25), tira-se

$$M\epsilon^{(k+1)} = N\epsilon^{(k)} \text{ ou } \epsilon^{(k+1)} = M^{-1}N\epsilon^{(k)}$$

cuja substituição recursiva origina

$$\epsilon^{(k+1)} = (M^{-1}N)^k \epsilon^{(1)}, \tag{3.26}$$

sendo $\epsilon^{(1)}$ o erro da aproximação inicial fornecida ao processo iterativo. Se os vectores z_i, $i = 1, \ldots, n$ formarem um conjunto de vectores próprios linearmente independentes da matriz $M^{-1}N$ e se λ_i forem os valores próprios que lhes correspondem, o vector $\epsilon^{(1)}$ pode exprimir-se como uma combinação linear dos z_i

$$\epsilon^{(1)} = \sum_{i=1}^{n} \alpha_i z_i,$$

em que os α_i são escalares e, de (3.26), obtém-se

$$\epsilon^{(k+1)} = \sum_{i=1}^{n} \lambda_i^k \alpha_i z_i.$$

Daqui se conclui que existe convergência, isto é, $\epsilon^{(k+1)} \to 0$ quando $k \to \infty$, a partir de qualquer aproximação inicial $x^{(1)}$, se e só se todos os valores próprios da matriz $M^{-1}N$, que são os $\lambda_i, i = 1, \ldots, n$, forem menores do que um, em valor absoluto.

À matriz $C = M^{-1}N$ chama-se *matriz de iteração*. Se pensarmos no valor próprio de C de maior módulo, em valor absoluto, e o designarmos por $\rho(C)$, *raio espectral*, então a condição necessária e suficiente de convergência do método iterativo baseado na equação (3.25) é

$$\rho(C) < 1; \tag{3.27}$$

e quanto mais próximo de um estiver $\rho(C)$, mais lenta é a convergência.

O raio espectral pode ser calculado usando o processo iterativo descrito em 5.3.1, (algoritmo 5.2) no Capítulo 5.

Dois processos para estimar limites superiores do raio espectral, $\rho(C)$, consideram as seguintes propriedades:

1. Todos os valores próprios de uma matriz, designadamente da matriz C, são, em valor absoluto, menores ou iguais a qualquer das normas dessa matriz e também

   $$\rho(C) \leq \|C\|_1, \|C\|_2, \|C\|_\infty.$$

   Assim, se a condição $\|C\| < 1$ se verificar, conclui-se que $\rho(C) < 1$ e verifica-se convergência do processo iterativo. Como esta condição é suficiente e não necessária, pode acontecer que as normas sejam maiores do que 1 embora os valores próprios sejam menores do que 1, donde resulta convergência. Ou, os valores próprios podem ser maiores do que um, donde resulta, neste caso, divergência. Portanto, se $\|C\| > 1$ nada se pode concluir sobre a convergência. É necessário calcular o raio espectral.

2. Todos os valores próprios de uma matriz, por exemplo da C, estão na união dos círculos,

   $$|\lambda(C) - c_{ii}| < \sum_{j=1, j\neq i}^{n} |c_{ij}|, \;\; i = 1, \ldots, n$$

   com centros c_{ii} e raios menores do que $\sum_{j=1,j\neq i}^{n} |c_{ij}|$. Determinado este conjunto, é possível verificar se todos os valores próprios de C são, em valor absoluto, menores do que um.

No que diz respeito aos métodos de Jacobi e Gauss-Seidel é possível estabelecer dois resultados importantes relativos à convergência para casos especiais de matrizes A (A é a matriz dos coeficientes do sistema $Ax = b$):

1. Se a matriz A é estrita e diagonalmente dominante, ao longo das linhas,

   $$|a_{ii}| > \sum_{j=1, j\neq i}^{n} |a_{ij}|, i = 1, \ldots, n,$$

   tanto o método de Jacobi como o de Gauss-Seidel convergem;

2. Se a matriz A é simétrica e definida positiva, o método de Gauss-Seidel converge. O método de Jacobi pode ou não convergir.

3.6.2 Equações de Jacobi

O *método iterativo de Jacobi*[6] usa a seguinte partição de A:

$$A = \mathcal{D} - (\mathcal{L} + \mathcal{U})$$

em que a matriz diagonal $\mathcal{D}$ é formada pelos elementos que estão na diagonal principal de A e $\mathcal{L}$ é formada pelos elementos que estão na parte estritamente triangular inferior de A, com os sinais trocados e todos os outros elementos são nulos. A matriz $\mathcal{U}$ é formada pelos elementos que estão na parte estritamente triangular superior de A, com os sinais trocados, sendo zeros todos os outros elementos. Como $\mathcal{D}$ tem de ser não singular, os $a_{ii}, i = 1, \ldots, n$ de A terão de ser diferentes de zero, para se poder aplicar o método de Jacobi. A correspondente matriz de iteração é

$$C_J = \mathcal{D}^{-1}(\mathcal{L} + \mathcal{U}).$$

De (3.25) resulta que a equação iterativa do método de Jacobi é

$$\mathcal{D}x^{(k+1)} = (\mathcal{L} + \mathcal{U})x^{(k)} + b \quad \text{ou} \quad x^{(k+1)} = \mathcal{D}^{-1}(\mathcal{L} + \mathcal{U})x^{(k)} + \mathcal{D}^{-1}b \tag{3.28}$$

para $k = 1, 2, 3, \ldots$. Em cada iteração, obtém-se um elemento do vector $x^{(k+1)}$, de cada vez, a partir do vector da iteração anterior $x^{(k)}$, isto é, o elemento $x_i^{(k+1)}$ é calculado da $i^{ésima}$ equação, a partir dos outros elementos do vector da iteração anterior,

$$x_i^{(k+1)} = \frac{-\sum_{j=1, j\neq i}^{n} a_{ij} x_j^{(k)} + b_i}{a_{ii}}, \text{ para } i = 1, \ldots, n.$$

A paragem deste processo iterativo deve verificar-se quando a aproximação calculada, $x^{(k+1)}$, estiver próxima da solução exacta. Uma maneira de se conseguir isto, consiste em estimar o erro relativo da aproximação, usando a quantidade

$$\frac{\|x^{(k+1)} - x^{(k)}\|}{\|x^{(k+1)}\|}$$

e exigir que esta seja igual ou inferior à precisão desejada para os resultados, por exemplo, ε.

O algoritmo 3.10 apresenta os passos correspondentes a este processo iterativo.

[6] C. G. Jacobi nasceu em 1804 e viveu grande parte da sua vida nas duas cidades alemãs de Königsber e Berlim. Foi um dos matemáticos notáveis do século 19 que, de uma certa maneira, deram continuação ao trabalho desenvolvido por Gauss. As suas publicações abrangem aspectos relacionados com a Análise real e complexa, a Teoria dos Números, a Mecânica e os sistemas de equações lineares. Este seu interesse foi despertado pelos trabalhos de Gauss no domínio dos mínimos quadrados. O seu nome está também ligado à integração numérica. Jacobi faleceu no ano de 1851. (Goldstine (1977) e Hämmerlin e Hoffmann (1991)).

Algoritmo 3.10 :

1. ler n, n_{max}, ε, calcular tol , para $i = 1, \ldots, n$ ler b_i e $(a_{ij}, j = 1, \ldots, n)$ e fazer $k = 0$
2. para $i = 1, \ldots, n$ fazer $x_i^{(1)} = 0$
3. para $i = 1, \ldots, n$ fazer
   - 3.1. se $|a_{ii}| \leq tol$ então terminar sem solução
   - 3.2. para $j = 1, \ldots, n$
     - 3.2.1. se $j = i$ então fazer $aux_{ii} = 0$
     - 3.2.2. senão calcular $aux_{ij} = -\frac{a_{ij}}{a_{ii}}$
   - 3.3. fazer $b_i = \frac{b_i}{a_{ii}}$
4. para $i = 1, \ldots, n$ calcular $soma_i = \sum_{j=1}^{n} |aux_{ij}|$
5. calcular $norma = \max(soma_1, soma_2, \ldots, soma_n)$
6. se $norma \geq 1$ então o processo iterativo pode não convergir
7. fazer $k = k + 1$
8. para $i = 1, \ldots, n$ calcular $x_i^{(k+1)} = \sum_{j=1}^{n} aux_{ij} x_j^{(k)} + b_i$
9. verificar o critério de paragem :
   - 9.1. calcular $norma2 = \sqrt{\sum_{i=1}^{n} (x_i^{(k+1)})^2}$
   - 9.2. calcular $norma1 = \sqrt{\sum_{i=1}^{n} (x_i^{(k+1)} - x_i^{(k)})^2}$
   - 9.3. se $norma2 \leq tol$ então fazer $cri = norma1$ senão fazer $cri = \frac{norma1}{norma2}$
   - 9.4. se $cri > \varepsilon$ então
     - 9.4.1. se $k \geq n_{max}$ então terminar, o processo não converge
     - 9.4.2. ir para o passo 7
10. terminar com $x^* \leftarrow (x_i^{(k+1)}, i = 1, \ldots, n)$.

3.6.3 Implementação. Rotina JACOBI

A implementação do algoritmo 3.10 é feita na rotina JACOBI que fornece os seguintes resultados:

Exemplo 3.16 Considerando o sistema

$$\begin{cases} 2x_1 + x_2 + x_3 &= 5 \\ 2x_1 + 3x_2 + x_3 &= 9 \\ x_1 + x_2 + 3x_3 &= 6 \end{cases}$$

e com $x^{(1)} = (0, 0, 0)^T$, o método converge ao fim de 129 iterações, com uma precisão, em termos relativos, igual a $\varepsilon = 10^{-6}$. Os vectores calculados ao longo das iterações foram os seguintes:

iteração, k	$x^{(k+1)}$	iteração, k	$x^{(k+1)}$
1	$(2.5, 3, 2)^T$	50	$(0.995014, 1.994869, 0.996224)^T$
2	$(0, 0.666667, 0.166667)^T$	51	$(1.004454, 2.004583, 1.003373)^T$
3	$(2.083333, 2.944444, 1.777778)^T$	100	$(0.999982, 1.999982, 0.999987)^T$
25	$(1.084078, 2.086519, 1.063669)^T$	129	$(1.000001, 2.000001, 1.000001)^T$

Exemplo 3.17 Considerando o sistema

$$\begin{cases} 6x_1 + x_2 + 2x_3 + x_5 = 10 \\ 2x_1 + 8x_2 + x_3 + 2x_4 + 2x_5 = 15 \\ x_1 - 2x_2 + 8x_3 + x_4 = 8 \\ -x_3 + 9x_4 + 2x_5 = 10 \\ x_1 + x_2 - x_4 + 7x_5 = 8 \end{cases}$$

e $x^{(1)} = (0, 0, 0, 0, 0)^T$, o método converge ao fim de 13 iterações, com uma precisão, em termos relativos, igual a 10^{-6} e a solução calculada é $x = (1.000000, 1.000000, 1.000000, 1.000000, 1.000000)^T$. A norma infinita da matriz de iteração é menor do que um.

3.6.4 Equações de Gauss-Seidel

Um outro método iterativo estacionário, também muito usado e de implementação muito simples é o *método iterativo de Gauss-Seidel*[7]. As duas matrizes da partição de A são agora

$$M = \mathcal{D} - \mathcal{L} \quad \text{e} \quad N = \mathcal{U}$$

em que $\mathcal{D}, \mathcal{L}$ e $\mathcal{U}$ são iguais às definidas para o método de Jacobi. Supondo que $\mathcal{D} - \mathcal{L}$ é uma matriz não singular, a matriz de iteração do método de Gauss-Seidel é

$$C_{GS} = (\mathcal{D} - \mathcal{L})^{-1}\mathcal{U}, \tag{3.29}$$

definindo-se para a equação iterativa

$$(\mathcal{D} - \mathcal{L})x^{(k+1)} = \mathcal{U}x^{(k)} + b \quad \text{ou} \quad x^{(k+1)} = (\mathcal{D} - \mathcal{L})^{-1}\mathcal{U}x^{(k)} + (\mathcal{D} - \mathcal{L})^{-1}b \tag{3.30}$$

para $k = 1, 2, 3, \ldots$. A diferença principal entre este processo iterativo e o de Jacobi reside no facto de que cada elemento, $x_i^{(k+1)}$, do vector, é calculado a partir da equação i, tendo como base apenas alguns elementos do vector da iteração anterior, $x_j^{(k)}, j > i$ e os outros elementos, são já do vector da presente iteração, $x_j^{(k+1)}, j < i$. Assim, para $i = 1, \ldots, n$ tem-se

$$x_i^{(k+1)} = \frac{-\sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)} + b_i}{a_{ii}}.$$

[7] L. Seidel foi um estudante de Jacobi e professor em Munique, na Alemanha. Interessou-se também pela técnica dos mínimos quadrados e pelos sistemas de equações lineares. O método iterativo, hoje conhecido por Gauss-Seidel, teve vários nomes associado a ele. Começou por Gauss, que o descreveu como método de aproximação, depois Gerling, em 1845, publicou-o num contexto de geometria, Jacobi, também em 1845, fez a apresentação de um método semelhante e finalmente Seidel publicou-o em 1874 (Goldstine (1977)).

O algoritmo que corresponde ao método de Gauss-Seidel está representado no Algoritmo 3.11.

Algoritmo 3.11 :

1. ler n, n_{max}, ε, calcular tol, para $i = 1, \ldots, n$ ler b_i e $(a_{ij}, j = 1, \ldots, n)$ e fazer $k = 0$
2. para $i = 1, \ldots, n$ fazer $x_i^{(1)} = 0$
3. calcular a inversa da matriz triangular inferior, $((a_{ij}, j = 1, \ldots, i), i = 1, \ldots, n)$: para $j = 1, \ldots, n$ fazer
   - 3.1. se $|a_{jj}| \leq tol$ então terminar sem solução
   - 3.2. para $i = j, \ldots, n$
     - 3.2.1. se $j = i$ então calcular $inv_{jj} = \frac{1}{a_{jj}}$
     - 3.2.2. senão calcular $inv_{ij} = \frac{\sum_{l=j}^{i-1}(-a_{il})inv_{lj}}{a_{ii}}$
4. para $i = 1, \ldots, n$ fazer
   - 4.1. para $j = 1, \ldots, n$
     - 4.1.1. se $j = 1$ então fazer $elem_{ij} = 0$
     - 4.1.2. senão
       - 4.1.2.1. se $j \leq i$ então calcular $elem_{ij} = \sum_{l=1}^{j-1} inv_{il}(-a_{lj})$
       - 4.1.2.2. senão calcular $elem_{ij} = \sum_{l=1}^{i} inv_{il}(-a_{lj})$
   - 4.2 calcular $aux_i = \sum_{l=1}^{i} inv_{il} b_l$
5. para $i = 1, \ldots, n$ calcular $soma_i = \sum_{j=1}^{n} |elem_{ij}|$
6. calcular $norma = \max(soma_1, soma_2, \ldots, soma_n)$
7. se $norma \geq 1$ então o processo iterativo pode não convergir
8. fazer $k = k + 1$
9. para $i = 1, \ldots, n$ calcular $x_i^{(k+1)} = \sum_{j=1}^{n} elem_{ij} x_j^{(k)} + aux_i$
10. verificar o critério de paragem :
    - 10.1. calcular $norma2 = \sqrt{\sum_{i=1}^{n} (x_i^{(k+1)})^2}$
    - 10.2. calcular $norma1 = \sqrt{\sum_{i=1}^{n} (x_i^{(k+1)} - x_i^{(k)})^2}$
    - 10.3. se $norma2 \leq tol$ então fazer $cri = norma1$ senão fazer $cri = \frac{norma1}{norma2}$
    - 10.4. se $cri > \varepsilon$ então
      - 10.4.1. se $k \geq n_{max}$ então terminar, o processo não converge
      - 10.4.2. ir para o passo 8
11. terminar com $x^* \leftarrow (x_i^{(k+1)}, i = 1, \ldots, n)$.

3.6.5 Implementação. Rotina GAUSEI

Na rotina GAUSEI está implementado o algoritmo 3.11.

Exemplo 3.18 Para o sistema do exemplo 3.16, a rotina GAUSEI fornece os seguintes resultados:

iteração, k	$x^{(k+1)}$	iteração, k	$x^{(k+1)}$
1	$(2.5, 1.333333, 0.722222)^T$	7	$(1.001833, 1.999086, 0.999694)^T$
3	$(1.152778, 1.925926, 0.973765)^T$	14	$(1.000001, 2.000000, 1.000000)^T$

para uma precisão igual à indicada no exemplo 3.16.

A matriz de iteração do método de Gauss-Seidel é

$$C_{GS} = \begin{pmatrix} 0.5 & 0 & 0 \\ -0.333333 & 0.333333 & 0 \\ -0.055556 & -0.111111 & 0.333333 \end{pmatrix} \begin{pmatrix} 0 & -1 & -1 \\ 0 & 0 & -1 \\ 0 & 0 & 0 \end{pmatrix}$$

$$= \begin{pmatrix} 0 & -0.5 & -0.5 \\ 0 & 0.333333 & 0 \\ 0 & 0.055556 & 0.166667 \end{pmatrix},$$

$\|C_{GS}\|_\infty = mx(1, 0.3, 0.222223) = 1$, $\|C_{GS}\|_1 = mx(0, 0.888889, 0.666667) = 0.888889$. Como $\|C_{GS}\|_1 < 1$, concluímos que o método de Gauss-Seidel converge para a solução deste sistema. Se quiséssemos calcular o raio espectral de C_{GS}, usaríamos o algoritmo 5.2 de 5.3.1 (Capítulo 5). O valor de $|\lambda_1(C_{GS})|$ $(= \rho(C_{GS}))$, obtido pela rotina MAIVAL, é de 0.330769.

Exemplo 3.19 Para o sistema do exemplo 3.17, a rotina GAUSEI apresenta a solução $x = (1.000000, 1.000000, 1.000000, 1.000000, 1.000000)^T$, ao fim de 9 iterações, considerando o parâmetro do critério de paragem, $\varepsilon = 10^{-6}$. A norma da matriz de iteração é também menor do que um.

3.7 Problemas

1. Estude o condicionamento do sistema $Ax = b$ sendo

$$A = \begin{pmatrix} 600 & 800 \\ 30001 & 40002 \end{pmatrix} \text{ e } b = \begin{pmatrix} 200 \\ 10001 \end{pmatrix}.$$

2. Considere o seguinte sistema de equações lineares

$$\begin{cases} -30x_1 + 9x_2 + 9x_3 &= 10 \\ 10x_1 - 2.99x_2 - 2.99x_3 &= 20 \\ 6x_1 - 6x_2 - 20x_3 &= 10 \end{cases}$$

   a) Estude o condicionamento deste sistema, justificando a sua resposta.

   b) Que outra técnica poderia usar para este estudo.

3. Dada a matriz

$$A = \begin{pmatrix} 2.4 & 6.0 & -2.7 & 5.0 \\ -2.1 & -2.7 & 5.9 & -4.0 \\ 3.0 & 5.0 & -4.0 & 6.0 \\ 0.9 & 1.9 & 4.7 & 1.8 \end{pmatrix}$$

e o vector $b = (14.6, -11.4, 14.0, -0.9)^T$,

a) Use um método directo e estável para condensar A à forma U.

b) Determine os factores triangulares de A.

c) Verifique se o determinante da matriz A é igual a -4.872.

4. Calcule a solução do sistema tridiagonal $Tx = b$ sendo

$$T = \begin{pmatrix} 4 & 1 & 0 & 0 \\ 2 & 4 & 1 & 0 \\ 0 & 2 & 4 & 1 \\ 0 & 0 & 2 & 4 \end{pmatrix}$$

e $b = (4, 1, -3, -1)^T$.

5. Calcule os factores triangulares da matriz

$$A = \begin{pmatrix} 2.4 & 6.0 & -2.7 & 5.0 \\ -2.1 & -2.7 & 5.9 & -4.0 \\ 3.0 & 5.0 & -4.0 & 6.0 \\ 0.9 & 1.9 & 4.7 & 1.8 \end{pmatrix}$$

usando um processo que conserve a estabilidade. Quantas vezes teve necessidade de trocar linhas? Qual o valor do determinante de A?

6. Calcule a factorização Cholesky da matriz

$$A = \begin{pmatrix} 1 & 0.5 & 0.5 \\ 0.5 & 1 & 0.5 \\ 0.5 & 0.5 & 1 \end{pmatrix}.$$

7. Mostre que a matriz

$$A = \begin{pmatrix} 2 & 4 & 6 \\ 4 & 7 & 8 \\ 6 & 8 & 5 \end{pmatrix}$$

não é definida positiva pela implementação do algoritmo 3.7.

8. Verifique que a inversa da matriz

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 10 \end{pmatrix} \text{ é } A^{-1} = \begin{pmatrix} 0.666667 & -1.333333 & 1.000000 \\ -0.666667 & 3.666667 & -2.000000 \\ 1.000000 & -2.000000 & 1.000000 \end{pmatrix}$$

com seis casas decimais.

9. Dada a matriz

$$A = \begin{pmatrix} 5 & -8 \\ -4 & 1 \end{pmatrix},$$

a) Verifique que os factores Q e R da factorização QR são respectivamente iguais a

$$\begin{pmatrix} -0.780868808 & 0.624695046 \\ 0.624695046 & 0.78086881 \end{pmatrix} \text{ e } \begin{pmatrix} -6.403124224 & 6.87164551 \\ & -4.216691558 \end{pmatrix}.$$

b) Calcule $det\,(A)$.

10. Verifique que o determinante da matriz

$$A = \begin{pmatrix} 0.1 & 0.2 & 0.3 \\ 0.4 & 0.5 & 0.6 \\ 0.7 & 0.8 & 0.9 \end{pmatrix}$$

é um valor próximo de zero. Estime o número de condição de A, calculando para isso a sua inversa. Que conclusões pode tirar em relação à sensibilidade da solução de um sistema linear que tivesse como matriz dos coeficientes a matriz A?

11. Considere o seguinte sistema de equações lineares

$$\begin{cases} 0.8x_1 + 1.4x_2 + 3.0x_3 &= 12.6 \\ 0.6x_1 + 0.9x_2 + 2.8x_3 &= 10.8 \\ 2.0x_1 + 1.0x_2 + 1.0x_3 &= 4.0 \end{cases}$$

Estude a convergência do método iterativo de Jacobi na sua resolução. Justifique.

12. Considere o seguinte sistema de equações lineares

$$\begin{cases} -30x_1 + 9x_2 + 9x_3 &= 10 \\ 10x_1 - 2.99x_2 - 2.99x_3 &= 20 \\ 6x_1 - 6x_2 - 20x_3 &= 10 \end{cases}$$

a) Verifique as condições suficientes de convergência do método iterativo de Gauss-Seidel, para a resolução deste sistema.
Que conclusões pode tirar sobre a convergência do método?

b) Resolva o sistema dado pelo método iterativo de Gauss-Seidel. Apresente os cálculos relativos às cinco primeiras iterações. Comente os resultados.

13. Verifique que a implementação do algoritmo 3.10 do método de Jacobi, para a resolução do sistema $Ax = b$, sendo

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 1/6 & 2/3 & 1/6 & 0 & 0 \\ 0 & 1/6 & 2/3 & 1/6 & 0 \\ 0 & 0 & 1/6 & 2/3 & 1/6 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

e $b = (0, -2, 1.5, -1, 1)^T$, leva 14 iterações para atingir a solução ($\varepsilon = 10^{-6}$)

$$x = (0.000000, -3.982141, 3.928569, -2.732141, 1.000000)^T.$$

14. Considere o sistema

$$\begin{cases} x_1 + 0.5x_2 + 0.5x_3 &= 2 \\ 0.5x_1 + x_2 + 0.5x_3 &= 2 \\ 0.5x_1 + 0.5x_2 + x_3 &= 2 \end{cases}$$

a) Use o método iterativo de Gauss-Seidel para calcular a solução, com uma precisão igual a 10^{-4}.

b) Calcule a norma infinita da matriz de iteração do método iterativo de Jacobi. Comente o valor obtido relativamente à sua convergência.

15. O sistema de equações lineares

$$\begin{pmatrix} 1 & -a \\ -a & 1 \end{pmatrix} x = b,$$

em que a é um valor real e $b, x \in R^2$, pode ser resolvido pelos métodos iterativos de Jacobi e Gauss-Seidel, desde que a verifique certas condições. Especifique essas condições para cada caso.

Capítulo 4

Sistemas de equações não lineares

4.1 Introdução

4.1.1 Forma geral do problema

Dado um conjunto de n funções não lineares, $f_1, f_2, \ldots, f_n$, nas n variáveis $x_1, x_2, \ldots, x_n$, pretende-se calcular o vector solução $x^* = (x_1^*, \ldots, x_n^*)^T$ que verifica simultaneamente as n equações do sistema

$$\begin{cases} f_1(x_1, x_2, \ldots, x_n) = 0 \\ f_2(x_1, x_2, \ldots, x_n) = 0 \\ \ldots \\ f_n(x_1, x_2, \ldots, x_n) = 0 \end{cases}$$

em que $f = (f_1, \ldots, f_n)^T$ é um vector de funções continuamente diferenciáveis (pelo menos até à primeira ordem).

4.1.2 Tipos de métodos e critério de paragem

Os métodos para a resolução do problema

$$f(x) = 0,$$

com $f : \mathrm{R}^n \to \mathrm{R}^n$ são iterativos. Comecemos pelo método de Newton e, como se verá pela dedução da equação iterativa, trata-se de uma extensão do método de Newton-Raphson apresentado em 2.8 do Capítulo 2. Um método alternativo ao de Newton é o método da secante de Broyden. Quando o cálculo analítico das derivadas se torna difícil e dispendioso, a matriz do Jacobiano, das primeiras derivadas das funções, deve ser aproximada por diferenças finitas ou actualizada usando fórmulas deduzidas a partir do método da secante. Tanto para o método de Newton como para o método da secante são analisadas questões relacionadas com a convergência, nomeadamente a razão de convergência, e com a paragem do processo iterativo.

Para a paragem do processo iterativo, o critério que deve ser usado baseia-se nas duas condições que a seguir se descrevem.

i) Na resolução de um sistema de equações não lineares, verifica-se se a aproximação $x^{(k+1)}$ resolve o problema, testando $f(x^{(k+1)}) = 0$. Em algoritmos iterativos e aritmética de precisão finita, a condição $f(x^{(k+1)}) = 0$ deve ser modificada para $f(x^{(k+1)}) \approx 0$. Assim, o teste mais adequado define

$$\|f(x^{(k+1)})\|_2 \leq \varepsilon_2 \tag{4.1}$$

desde que as equações estejam razoavelmente bem escalonadas entre si.

ii) Também se verifica se o algoritmo convergiu, ou se estabilizou, no ponto $x^{(k+1)}$, se a alteração relativa verificada em duas iterações consecutivas, $k+1$ e k, pode ser considerada desprezável, isto é, se

$$\frac{\|\Delta_x\|_2}{\|x^{(k+1)}\|_2} \leq \varepsilon_1, \tag{4.2}$$

em que

$$x^{(k+1)} = x^{(k)} + \Delta_x.$$

ε_1 e ε_2 são duas quantidades positivas e próximas de zero. Se o vector $x^{(k+1)}$ está a convergir para zero ou passa por valores próximos de zero, a condição (4.2) deve ser substituída por

$$\|\Delta_x\|_2 \leq \varepsilon_1.$$

Em muitos processos iterativos é possível quantificar o gasto excessivo dos recursos, nomeadamente o tempo, impondo um limite nas iterações efectuadas. Este limite depende do custo computacional de cada iteração e do problema a resolver. Assim, a condição que deve ser incluída, não permitindo que se façam mais iterações do que um valor pré-especificado é,

$$\text{terminar} \quad \text{se} \quad k \geq n_{max}.$$

O algoritmo 4.1 corresponde ao critério de paragem baseado nas condições (4.1) e (4.2).

Algoritmo 4.1 :

1. `ler` n, k, ε_1 , ε_2 , `para` $j = 1, \ldots, n$ `ler` $x_j^{(k+1)}$ `e` Δ_{x_j} `e para` $i = 1, \ldots, n$ `ler` $f_i(x^{(k+1)})$ `e calcular` tol
2. `calcular` $norma1 = \sqrt{\sum_{j=1}^{n} \Delta_{x_j}^2}$
3. `calcular` $norma2 = \sqrt{\sum_{j=1}^{n} (x_j^{(k+1)})^2}$
4. `calcular` $norma3 = \sqrt{\sum_{i=1}^{n} f_i(x^{(k+1)})^2}$
5. `se` $norma2 \leq tol$ `então fazer` $cri = norma1$ `senão fazer` $cri = \frac{norma1}{norma2}$
6. `se` $cri \leq \varepsilon_1$ `e` $norma3 \leq \varepsilon_2$ `então terminar com convergência=.TRUE.`
7. `terminar com convergência=.FALSE.`

4.1.3 Índice de algoritmos

Neste Capítulo são apresentados três algoritmos.

Algoritmo 4.1 Critério de paragem dos métodos iterativos

Algoritmo 4.2 Resolução de um sistema não linear pelo método iterativo de Newton

Algoritmo 4.3 Resolução de um sistema não linear por um método do tipo secante

O primeiro descreve o critério de paragem de um processo iterativo para o cálculo da solução de um sistema de equações não lineares, o algoritmo 4.2 descreve o método de Newton e que deve ser usado quando estiverem disponíveis as primeiras derivadas das funções. O algoritmo 4.3 é do tipo secante e é baseado na equação de Broyden para actualização da matriz do Jacobiano.

4.2 Método de Newton

4.2.1 Introdução teórica. Convergência

Considerando a expansão em série de Taylor de $f(x)$, em relação ao ponto x,

$$\begin{aligned}
f_1(\bar{x}_1,\ldots,\bar{x}_n) &= f_1(x_1,\ldots,x_n) + (\bar{x}_1 - x_1)\frac{\partial f_1(x_1,\ldots,x_n)}{\partial x_1} + (\bar{x}_2 - x_2)\frac{\partial f_1(x_1,\ldots,x_n)}{\partial x_2} \\
&+ \cdots + (\bar{x}_n - x_n)\frac{\partial f_1(x_1,\ldots,x_n)}{\partial x_n} + \cdots \\
f_2(\bar{x}_1,\ldots,\bar{x}_n) &= f_2(x_1,\ldots,x_n) + (\bar{x}_1 - x_1)\frac{\partial f_2(x_1,\ldots,x_n)}{\partial x_1} + (\bar{x}_2 - x_2)\frac{\partial f_2(x_1,\ldots,x_n)}{\partial x_2} \\
&+ \cdots + (\bar{x}_n - x_n)\frac{\partial f_2(x_1,\ldots,x_n)}{\partial x_n} + \cdots \\
&\cdots \\
f_n(\bar{x}_1,\ldots,\bar{x}_n) &= f_n(x_1,\ldots,x_n) + (\bar{x}_1 - x_1)\frac{\partial f_n(x_1,\ldots,x_n)}{\partial x_1} + (\bar{x}_2 - x_2)\frac{\partial f_n(x_1,\ldots,x_n)}{\partial x_2} \\
&+ \cdots + (\bar{x}_n - x_n)\frac{\partial f_n(x_1,\ldots,x_n)}{\partial x_n} + \cdots
\end{aligned}$$

e tomando apenas os $n+1$ primeiros termos de cada equação, ignorando os restantes, obtém-se um modelo linear, $l(\bar{x})$, aproximação a $f(\bar{x})$,

$$\begin{aligned}
l_1(\bar{x}_1,\ldots,\bar{x}_n) &= f_1(x_1,\ldots,x_n) + (\bar{x}_1 - x_1)\frac{\partial f_1(x_1,\ldots,x_n)}{\partial x_1} + (\bar{x}_2 - x_2)\frac{\partial f_1(x_1,\ldots,x_n)}{\partial x_2} \\
&+ \cdots + (\bar{x}_n - x_n)\frac{\partial f_1(x_1,\ldots,x_n)}{\partial x_n} \\
l_2(\bar{x}_1,\ldots,\bar{x}_n) &= f_2(x_1,\ldots,x_n) + (\bar{x}_1 - x_1)\frac{\partial f_2(x_1,\ldots,x_n)}{\partial x_1} + (\bar{x}_2 - x_2)\frac{\partial f_2(x_1,\ldots,x_n)}{\partial x_2}
\end{aligned}$$

$$
\begin{aligned}
& + \quad \cdots + (\bar{x}_n - x_n)\frac{\partial f_2(x_1,\ldots,x_n)}{\partial x_n} \\
& \quad \cdots \\
l_n(\bar{x}_1,\ldots,\bar{x}_n) &= f_n(x_1,\ldots,x_n) + (\bar{x}_1 - x_1)\frac{\partial f_n(x_1,\ldots,x_n)}{\partial x_1} + (\bar{x}_2 - x_2)\frac{\partial f_n(x_1,\ldots,x_n)}{\partial x_2} \\
& + \quad \cdots + (\bar{x}_n - x_n)\frac{\partial f_n(x_1,\ldots,x_n)}{\partial x_n}.
\end{aligned}
$$

O vector $\bar{x}$ que resolve o sistema $l(\bar{x}) = 0$ calcula-se, então, a partir de

$$
\begin{pmatrix} \bar{x}_1 \\ \bar{x}_2 \\ \cdots \\ \bar{x}_n \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \\ \cdots \\ x_n \end{pmatrix} + \begin{pmatrix} \Delta_{x_1} \\ \Delta_{x_2} \\ \cdots \\ \Delta_{x_n} \end{pmatrix},
$$

em que o vector $\Delta_x \in \mathbf{R}^n$ é a solução do seguinte sistema de equações lineares:

$$
\begin{pmatrix}
\frac{\partial f_1(x_1,\ldots,x_n)}{\partial x_1} & \frac{\partial f_1(x_1,\ldots,x_n)}{\partial x_2} & \cdots & \frac{\partial f_1(x_1,\ldots,x_n)}{\partial x_n} \\
\frac{\partial f_2(x_1,\ldots,x_n)}{\partial x_1} & \frac{\partial f_2(x_1,\ldots,x_n)}{\partial x_2} & \cdots & \frac{\partial f_2(x_1,\ldots,x_n)}{\partial x_n} \\
\cdot & \cdot & & \cdot \\
\cdot & \cdot & & \cdot \\
\cdot & \cdot & & \cdot \\
\frac{\partial f_n(x_1,\ldots,x_n)}{\partial x_1} & \frac{\partial f_n(x_1,\ldots,x_n)}{\partial x_2} & \cdots & \frac{\partial f_n(x_1,\ldots,x_n)}{\partial x_n}
\end{pmatrix} \Delta_x = - \begin{pmatrix} f_1(x_1,\ldots,x_n) \\ f_2(x_1,\ldots,x_n) \\ \cdot \\ \cdot \\ \cdot \\ f_n(x_1,\ldots,x_n) \end{pmatrix}.
$$

A matriz dos coeficientes deste sistema chama-se *matriz do Jacobiano* e é formada pelas primeiras derivadas parciais das funções $f_1, f_2, \ldots, f_n$ em ordem às variáveis $x_1, \ldots, x_n$. A partir de agora usa-se a letra J para identificar esta matriz.

Como o modelo $l(x) = 0$ é apenas uma aproximação linear à equação $f(x) = 0$, o vector $\bar{x}$ é apenas uma aproximação à solução x^*. O processo deve ser repetido, tomando como ponto de partida o vector $\bar{x}$, isto é, fazendo $x \leftarrow \bar{x}$, e calculando uma nova aproximação $\bar{x}$.

Para simplificar a notação usámos o vector x no lugar da aproximação da iteração k, $x^{(k)}$ e o vector $\bar{x}$ no lugar da aproximação da iteração seguinte: $x^{(k+1)}$. Assim, o processo iterativo do *método de Newton* consiste em calcular uma nova aproximação $x^{(k+1)}$, a partir da anterior $x^{(k)}$,

$$
x^{(k+1)} = x^{(k)} + \Delta_x, \ k = 1, 2, \ldots
$$

sendo o vector Δ_x calculado através da resolução do sistema das equações Newton, cuja matriz dos coeficientes é o Jacobiano J e o vector $-f$ é o termo independente, do lado direito, ambos calculados no ponto $x^{(k)}$,

$$
J(x^{(k)})\Delta_x = -f(x^{(k)}). \tag{4.3}
$$

Como a solução encontrada resolve exactamente o modelo linear, $l(x) = 0$, usado para aproximar $f(x) = 0$, $x^{(k+1)}$ não é a solução exacta de $f(x) = 0$ e o processo deve ser repetido iterativamente. Se estiver a convergir, as iterações do método de Newton devem ser terminadas quando x estabilizar e $f(x) \approx 0$, isto é, quando o critério de paragem descrito em 4.1.2 for verificado. A descrição algorítmica deste processo iterativo encontra-se no algoritmo 4.2.

Algoritmo 4.2 :

1. ler n e n_{max}
2. introduzir as funções $f_1(x_1, x_2, \ldots, x_n), f_2(x_1, x_2, \ldots, x_n), \ldots, f_n(x_1, x_2, \ldots, x_n)$ e as que compõem o Jacobiano:

$$\begin{pmatrix} \frac{\partial f_1(x_1,\ldots,x_n)}{\partial x_1} & \frac{\partial f_1(x_1,\ldots,x_n)}{\partial x_2} & \cdots & \frac{\partial f_1(x_1,\ldots,x_n)}{\partial x_n} \\ \frac{\partial f_2(x_1,\ldots,x_n)}{\partial x_1} & \frac{\partial f_2(x_1,\ldots,x_n)}{\partial x_2} & \cdots & \frac{\partial f_2(x_1,\ldots,x_n)}{\partial x_n} \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \frac{\partial f_n(x_1,\ldots,x_n)}{\partial x_1} & \frac{\partial f_n(x_1,\ldots,x_n)}{\partial x_2} & \cdots & \frac{\partial f_n(x_1,\ldots,x_n)}{\partial x_n} \end{pmatrix}$$

3. fazer $k = 0$

   3.1. para $j = 1, \ldots, n$ ler $x_j^{(1)}$

   3.2. para $i = 1, \ldots, n$ calcular $f_i(x_1^{(1)}, x_2^{(1)}, \ldots, x_n^{(1)})$

4. fazer $k = k + 1$
5. para $i = 1, \ldots, n$ e $j = 1, \ldots, n$ calcular

$$\frac{\partial f_i(x_1^{(k)}, x_2^{(k)}, \ldots, x_n^{(k)})}{\partial x_j}$$

6. resolver o sistema linear (algoritmo 3.3), em ordem ao vector Δ_x:

$$\begin{pmatrix} \frac{\partial f_1(x^{(k)})}{\partial x_1} & \frac{\partial f_1(x^{(k)})}{\partial x_2} & \cdots & \frac{\partial f_1(x^{(k)})}{\partial x_n} \\ \frac{\partial f_2(x^{(k)})}{\partial x_1} & \frac{\partial f_2(x^{(k)})}{\partial x_2} & \cdots & \frac{\partial f_2(x^{(k)})}{\partial x_n} \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \frac{\partial f_n(x^{(k)})}{\partial x_1} & \frac{\partial f_n(x^{(k)})}{\partial x_2} & \cdots & \frac{\partial f_n(x^{(k)})}{\partial x_n} \end{pmatrix} \Delta_x = - \begin{pmatrix} f_1(x^{(k)}) \\ f_2(x^{(k)}) \\ \cdot \\ \cdot \\ \cdot \\ f_n(x^{(k)}) \end{pmatrix}$$

7. para $j = 1, \ldots, n$ calcular $x_j^{(k+1)} = x_j^{(k)} + \Delta_{x_j}$
8. para $i = 1, \ldots, n$ calcular $f_i(x_1^{(k+1)}, x_2^{(k+1)}, \ldots, x_n^{(k+1)})$
9. se convergência=.FALSE. (algoritmo 4.1) então

`9.1. se` $k \geq n_{max}$ `então recomeçar, ir para o passo 3`

`9.2. ir para o passo 4`

`10. terminar com` $x^* \leftarrow (x_j^{(k+1)}, j = 1, \ldots, n)$.

O teorema seguinte mostra que a convergência do método de Newton é local e de ordem dois.

Teorema 4.1 : Seja $f : \mathbb{R}^n \to \mathbb{R}^n$ *uma função continuamente diferenciável num conjunto aberto e convexo* $D \subset \mathbb{R}^n$. *Suponha que existem* $x^* \in \mathbb{R}^n$, r *e* β *escalares positivos, tais que a vizinhança aberta de raio* r *e centro* x^*, $V(x^*, r) \subset D$,

$$f(x^*) = 0,$$

$$J(x^*) \text{ é não singular }, \ \|J(x^*)^{-1}\| \leq \beta$$

e $J(x)$ *é uma matriz Lipschitz contínua na vizinhança de* x^*, *em que a constante de Lipschitz é* γ. *Então existe uma quantidade positiva* ε *tal que, para todo o* $x^{(1)} \in V(x^*, \varepsilon)$, *a sequência* $\{x^{(k)}\}$ *gerada pela equação Newton*

$$x^{(k+1)} = x^{(k)} + \Delta_x$$

e

$$J(x^{(k)})\Delta_x = -f(x^{(k)}), \ k = 1, 2, \ldots$$

é bem definida, converge para x^* *e verifica*

$$\|x^{(k+1)} - x^*\| \leq \beta\gamma\|x^{(k)} - x^*\|^2, \ k = 1, 2, \ldots \tag{4.4}$$

Deve-se escolher ε de tal forma que $J(x)$ seja não singular para todo o $x \in V(x^*, \varepsilon)$, isto é,

$$\varepsilon = min\{r, \frac{1}{2\beta\gamma}\}.$$

A prova pode ser feita por indução. Em cada iteração, é preciso verificar que $J(x^{(k)})$ é não singular, $x^{(k+1)} \in V(x^*, \varepsilon)$ e a desigualdade (4.4).

Comecemos com $k = 1$ e por $J(x^{(1)})$. Como J é Lipschitz contínua e $\|x^{(1)} - x^*\| \leq \varepsilon$,

$$\begin{aligned} \|J(x^*)^{-1}[J(x^{(1)}) - J(x^*)]\| &\leq \|J(x^*)^{-1}\|\|J(x^{(1)}) - J(x^*)\| \\ &\leq \beta\gamma\|x^{(1)} - x^*\| \\ &\leq \beta\gamma\varepsilon \leq \frac{1}{2}. \end{aligned}$$

Através da relação

$$\|J(x^{(1)})^{-1}\| \leq \frac{\|J(x^*)^{-1}\|}{1 - \|J(x^*)^{-1}[J(x^{(1)}) - J(x^*)]\|}$$

e usando o Teorema 3.2, de 3.2.2 do Capítulo 3, com $B = J(x^*)^{-1}J(x^{(1)}) - I$, tem-se que $J(x^{(1)})$ é não singular e

$$\|J(x^{(1)})^{-1}\| \leq 2\|J(x^*)^{-1}\| \leq 2\beta,$$

o que implica que $x^{(2)}$ é bem definido. Como

$$\begin{aligned} x^{(2)} - x^* &= x^{(1)} - x^* - J(x^{(1)})^{-1}f(x^{(1)}) \\ &= x^{(1)} - x^* - J(x^{(1)})^{-1}[f(x^{(1)}) - f(x^*)] \\ &= J(x^{(1)})^{-1}[J(x^{(1)})(x^{(1)} - x^*) - f(x^{(1)}) + f(x^*)] \end{aligned}$$

tem-se

$$\begin{aligned} \|x^{(2)} - x^*\| &\leq \|J(x^{(1)})^{-1}\|\|-J(x^{(1)})(x^* - x^{(1)}) - f(x^{(1)}) + f(x^*)\| \\ &\leq 2\beta\|f(x^*) - l(x^*)\| \\ &\leq 2\beta\frac{\gamma}{2}\|x^{(1)} - x^*\|^2, \end{aligned}$$

o que prova a condição (4.4). Como $\|x^{(1)} - x^*\| \leq \varepsilon \leq \frac{1}{2\beta\gamma}$ tem-se

$$\|x^{(2)} - x^*\| \leq \frac{1}{2}\|x^{(1)} - x^*\|$$

o que quer dizer que $x^{(2)} \in V(x^*, \varepsilon)$. O resto da prova por indução segue os mesmos passos.

A razão de convergência do método de Newton, embora local, é uma das suas vantagens. O processo que originou as equações Newton, faz com que o método encontre a solução exacta de um sistema linear, ao fim de uma iteração. A necessidade de, em cada iteração, calcular a solução do sistema de equações (4.3), bem como o cálculo da matriz do Jacobiano, são os grandes inconvenientes que surgem na implementação do método de Newton. Em certas aplicações, as funções f não são conhecidas analiticamente e o Jacobiano não pode ser calculado. Para estes casos sugerem-se outros métodos baseados nas equações Newton, em que as derivadas são aproximadas por diferenças finitas (veja-se em 4.2.4). Outra alternativa, ainda mais simples e económica, em termos de cálculos computacionais, é a que se baseia na aproximação secante do Jacobiano. Falaremos do método do tipo secante em 4.3.

4.2.2 Implementação. Rotina NEWSIS

A rotina NEWSIS implementa o algoritmo 4.2. Para o critério de paragem foram usados os seguintes valores $\varepsilon_1 = \varepsilon_2 = 10^{-6}$.

Exemplo 4.1 Considere o seguinte sistema

$$\begin{cases} sen(x_1x_2) - x_2 + x_1 = 0 \\ x_2cos(x_1x_2) + 1 = 0 \end{cases}$$

e usando $(1,2)^T$ como vector inicial, a rotina NEWSIS fornece a solução $(1.086187, 1.943685)^T$ ao fim de 4 iterações. A matriz do Jacobiano tem o seguinte aspecto:

$$\begin{pmatrix} x_2 cos(x_1 x_2) + 1 & x_1 cos(x_1 x_2) - 1 \\ -x_2^2 sen(x_1 x_2) & cos(x_1 x_2) - x_1 x_2 sen(x_1 x_2) \end{pmatrix}$$

.

Exemplo 4.2 Para o sistema

$$\begin{cases} x_1^2 + x_2^2 - 9 = 0 \\ x_1 + x_2 - 1 = 0 \end{cases}$$

a implementação do algoritmo 4.2 leva 5 iterações para chegar ao vector $(2.561553, -1.561553)^T$. O vector inicial foi $(2,0)^T$. Esta solução não é a única. Existem dois pontos na intersecção da circunferência, centrada em $(0,0)$ e de raio 3, com a recta $x_1 + x_2 - 1 = 0$. Se iniciasse o processo iterativo com o vector $(0,0)^T$, o Jacobiano era singular e o sistema Newton não teria solução.

Exemplo 4.3 Finalmente para o sistema

$$\begin{cases} x_1 x_2 - x_2^3 - 1 = 0 \\ x_1^2 x_2 + x_2 - 5 = 0 \end{cases}$$

e a partir de $(2,3)^T$, obtém-se a solução $(2,1)^T$ ao fim de 8 iterações. Algumas iterações calculadas no processo:

k	$x^{(k+1)}$	k	$x^{(k+1)}$
1	$(1.555556, 2.066667)^T$	5	$(1.997763, 1.001240)^T$
2	$(1.547205, 1.477793)^T$	6	$(1.999995, 1.000003)^T$
3	$(1.780535, 1.158865)^T$	7	$(2.000000, 1.000000)^T$

4.2.3 Aplicação à determinação de raízes complexas

Uma das aplicações mais importantes do problema de equações não lineares a duas dimensões consiste na determinação das raízes complexas de uma equação da forma

$$f(z) = 0$$

em que $z = x_1 + x_2 \imath$ e $\imath$ é a unidade imaginária. Se $f(z) = f(x_1 + x_2\imath) = g(x_1, x_2) + h(x_1, x_2)\imath$, em que $g(x_1, x_2)$ é a parte real e $h(x_1, x_2)$ é a parte imaginária do complexo $f(z)$ e g, $h : \mathbf{R}^2 \to \mathbf{R}$, então, a resolução de $f(z) = 0$ é equivalente à resolução do sistema

$$\begin{cases} g(x_1, x_2) &= 0 \\ h(x_1, x_2) &= 0 \end{cases} \tag{4.5}$$

uma vez que um número complexo é nulo quando forem nulas as suas partes real e imaginária.

Considerando as equações de Cauchy-Riemann, relativas às derivadas parciais das funções $g(x_1, x_2)$ e $h(x_1, x_2)$ de um complexo, em ordem a x_1 e x_2 (respectivamente parte real e imaginária de z),

$$\frac{\partial g(x_1, x_2)}{\partial x_1} = \frac{\partial h(x_1, x_2)}{\partial x_2}$$

$$\frac{\partial h(x_1, x_2)}{\partial x_1} = -\frac{\partial g(x_1, x_2)}{\partial x_2}$$

obtém-se, da aplicação do sistema das equações Newton (4.3) ao sistema (4.5), o seguinte

$$\Delta_{x_1} = -(g\frac{\partial g}{\partial x_1} + h\frac{\partial h}{\partial x_1})/((\frac{\partial g}{\partial x_1})^2 + (\frac{\partial h}{\partial x_1})^2)$$

e

$$\Delta_{x_2} = -(h\frac{\partial g}{\partial x_1} - g\frac{\partial h}{\partial x_1})/((\frac{\partial g}{\partial x_1})^2 + (\frac{\partial h}{\partial x_1})^2).$$

As funções g e h, e as derivadas parciais $\frac{\partial g}{\partial x_1}$ e $\frac{\partial h}{\partial x_1}$ são calculadas no ponto da iteração corrente $(x_1^{(k)}, x_2^{(k)})^T$. A nova aproximação à solução é então calculada a partir de

$$\begin{pmatrix} x_1^{(k+1)} \\ x_2^{(k+1)} \end{pmatrix} = \begin{pmatrix} x_1^{(k)} \\ x_2^{(k)} \end{pmatrix} + \begin{pmatrix} \Delta_{x_1} \\ \Delta_{x_2} \end{pmatrix}.$$

Exemplo 4.4 Substituindo $z = x_1 + x_2\imath$ na equação $f(z) = z^3 + (1 - 2\imath)z^2 + 2z - 4 - 5\imath = 0$ obtém-se

$$(x_1^3 - 3x_1x_2^2 + x_1^2 - x_2^2 + 4x_1x_2 + 2x_1 - 4) + (-x_2^3 + 3x_1^2x_2 - 2x_1^2 + 2x_2^2 + 2x_1x_2 + 2x_2 - 5)\imath = 0$$

ou

$$\begin{cases} x_1^3 - 3x_1x_2^2 + x_1^2 - x_2^2 + 4x_1x_2 + 2x_1 - 4 &= 0 \\ -x_2^3 + 3x_1^2x_2 - 2x_1^2 + 2x_2^2 + 2x_1x_2 + 2x_2 - 5 &= 0 \end{cases}$$

Para $(x_1, x_2) = (0, 0)$ tem-se ao fim de uma iteração do método de Newton a aproximação $(2, 2.5)$, ou seja $z = 2 + 2.5\imath$.

4.2.4 Método baseado nas diferenças finitas

Uma das dificuldades que podem surgir durante a implementação do método de Newton está relacionada com a determinação analítica do Jacobiano. Esta questão é facilmente ultrapassada substituindo o Jacobiano, $J(x)$, por uma aproximação baseada em diferenças finitas,

$$J(x) \longleftarrow DF(x).$$

Assim, se os elementos da matriz do Jacobiano forem

$$J_{i,j} = \frac{\partial f_i(x_1, \ldots, x_n)}{\partial x_j}, \text{ com } i = 1, \ldots, n, \ j = 1, \ldots, n$$

tem-se

$$J_{i,j} \approx DF_{i,j} = \frac{f_i(x + h_j e_j) - f_i(x)}{h_j}, \tag{4.6}$$

em que $h_j \in \mathbf{R}$ é o passo e $e_j \in \mathbf{R}^n$ é o vector que define a coordenada j do espaço, isto é, a coluna j da matriz identidade de ordem n. Se a escolha do passo h_j for adequada, a convergência quadrática do método de Newton (Teorema 4.1) mantém-se. Esta escolha deve ter em consideração a aritmética discreta (de precisão finita).

Teorema 4.2 : Se f e x^* verificam as condições do Teorema 4.1, em que $\|.\|_1$ designa a norma 1 de vectores e a correspondente norma de matrizes, então existem quantidades $\varepsilon, h > 0$ tais que, se $\{h^{(k)}\}$ é uma sequência de números reais com $0 < |h^{(k)}| \leq h$ e $x^{(1)} \in V(x^*, \varepsilon)$, a sequência $x^{(2)}, x^{(3)}, \ldots$ gerada por

$$x^{(k+1)} = x^{(k)} + \Delta_x$$

e

$$DF(x^{(k)})\, \Delta_x = -f(x^{(k)}), \;\; k = 1, 2, \ldots$$

com

$$(DF(x^{(k)}))_{.j} = \frac{f(x^{(k)} + h^{(k)} e_j) - f(x^{(k)})}{h^{(k)}}, \;\; j = 1, \ldots, n$$

(coluna j da matriz de aproximação ao Jacobiano) é bem definida e converge linearmente para x^*.

Se $\lim_{k\to\infty} h^{(k)} = 0$ então a convergência é superlinear.

Se existir alguma constante c_1 tal que

$$|h^{(k)}| \leq c_1 \|x^{(k)} - x^*\|$$

ou, uma constante c_2 tal que

$$|h^{(k)}| \leq c_2 \|f(x^{(k)})\|,$$

então a convergência é quadrática.

Este teorema não dá indicações precisas sobre os valores a atribuir ao passo h. O teorema assume um valor constante em cada iteração, k. Este procedimento pode gerar dificuldades, especialmente se os elementos do vector x tiverem grandezas muito distintas. Considerando comprimentos de passo individuais diferentes, $h_j, j = 1, \ldots, n$, no teorema 4.2 obtêm-se as mesmas conclusões.

Para chegarmos a uma expressão adequada para h_j, temos de considerar as seguintes condicionantes:

- A aproximação às derivadas, por diferenças finitas, origina um erro de truncatura da ordem de h_j, em aritmética exacta. Isto significa que o passo deve ser escolhido tão pequeno quanto possível.

- O outro tipo de erros que se comete quando se calcula o numerador de (4.6), em aritmética discreta, é tanto maior quanto menor for h_j. Para valores de h_j muito pequenos, $x^{(k)} + h_j e_j \approx x^{(k)}$, $f(x^{(k)} + h_j e_j) \approx f(x^{(k)})$ e ocorre uma perda excessiva de algarismos significativos (situação já referida no Capítulo 1, em 1.5.1).

O valor de h_j deve ser escolhido por forma a verificar-se uma diferença nos últimos $t/2$ algarismos de $f(x^{(k)} + h_j e_j)$, quando comparado com $f(x^{(k)})$, se este tiver t algarismos significativos. Assim, se o erro relativo no cálculo de $f(x^{(k)})$ é η,

$$\frac{\|f(x^{(k)} + h_j e_j) - f(x^{(k)})\|}{\|f(x^{(k)})\|} \leq \sqrt{\eta}, \quad j = 1, \ldots, n$$

e a perturbação de cada elemento, x_j, do vector $x^{(k)}$ deve ser proporcional à sua grandeza e corresponder, em termos relativos, a $\sqrt{\eta}$ de x_j,

$$h_j = \sqrt{\eta}\, x_j. \tag{4.7}$$

Quando x_j está muito próximo de zero, a expressão (4.7) não é adequada. Em vez desta, deve ser usada a seguinte:

$$h_j = \sqrt{\eta}\, mx\{|x_j|, tip\, x_j\}\, sinal(x_j).$$

A variável $tip\, x_j$ indica o valor típico de x_j e traduz o valor médio dos valores prováveis de x_j, ao longo do processo iterativo. Se este valor for difícil de estimar, considera-se a unidade em vez dele.

Um valor apropriado para η deve traduzir a precisão do computador usado nos cálculos. Assim, sugere-se o uso da quantidade tol, introduzida pela primeira vez no Capítulo 2 e que pode ser calculada implementando o algoritmo 2.5.

4.3 Método do tipo secante

4.3.1 Introdução teórica. Equações de Broyden

Vamos introduzir uma extensão do método da secante para a resolução de sistemas de equações não lineares. Considere o modelo linear, aproximação a $f(x_1, \ldots, x_n)$,

$$l(x_1, \ldots, x_n) = f(x_1^{(k+1)}, \ldots, x_n^{(k+1)}) + A^{(k+1)}(x - x^{(k+1)}) \tag{4.8}$$

em que $l : \mathbf{R}^n \to \mathbf{R}^n$, $A^{(k+1)} \in \mathbf{R}^{n \times n}$, $x = (x_1, \ldots, x_n)^T$ e $x^{(k+1)} = (x_1^{(k+1)}, \ldots, x_n^{(k+1)})^T$, que satisfaz $l(x_1^{(k+1)}, \ldots, x_n^{(k+1)}) = f(x_1^{(k+1)}, \ldots, x_n^{(k+1)})$ para qualquer matriz $A^{(k+1)}$.

No método de Newton, em vez da matriz $A^{(k+1)}$, tem-se o Jacobiano calculado no ponto $x^{(k+1)}$, $J(x_1^{(k+1)}, \ldots, x_n^{(k+1)})$. Quando este não é conhecido, a matriz $A^{(k+1)}$ deve ser tal que

$$l(x_1^{(k)}, \ldots, x_n^{(k)}) = f(x_1^{(k)}, \ldots, x_n^{(k)})$$

sendo $(x_1^{(k)}, \ldots, x_n^{(k)})^T$ o ponto da iteração corrente, k. Assim,

$$f(x^{(k)}) = f(x^{(k+1)}) + A^{(k+1)}(x^{(k)} - x^{(k+1)})$$

ou

$$\begin{aligned} A^{(k+1)}(x^{(k+1)} - x^{(k)}) &= f(x^{(k+1)}) - f(x^{(k)}) && (4.9) \\ A^{(k+1)}\Delta_x &= y^{(k)}, && (4.10) \end{aligned}$$

equação conhecida por *condição secante*, em que $\Delta_x = x^{(k+1)} - x^{(k)}$ é o deslocamento efectuado e $y^{(k)} = f(x^{(k+1)}) - f(x^{(k)})$ é a variação no resultado depois de efectuado aquele deslocamento.

A equação (4.10) não especifica um valor único para $A^{(k+1)}$. Apenas n elementos desta matriz podem ser determinados em função dos restantes $n(n-1)$. Estes podem tomar quaisquer valores. Assim, para $\Delta_x \neq 0$, existe um subespaço linear de dimensão $n(n-1)$ de matrizes que satisfazem (4.10). Para seleccionar uma delas, exige-se que a matriz satisfaça outra condição. Vamos tentar minimizar a variação que surge no modelo linear, quando passamos do ponto $x^{(k)}$ para $x^{(k+1)}$, sujeita a (4.10). Para qualquer $x \in \mathbf{R}^n$, a diferença entre o modelo $l(x)$ antigo, na equação (4.8), e o modelo $l^{(k)}(x) = f(x^{(k)}) + A^{(k)}(x - x^{(k)})$ é dada por

$$\begin{aligned} l(x) - l^{(k)}(x) &= f(x^{(k+1)}) + A^{(k+1)}(x - x^{(k+1)}) - f(x^{(k)}) - A^{(k)}(x - x^{(k)}) \\ &= f(x^{(k+1)}) - f(x^{(k)}) - A^{(k+1)}(x^{(k+1)} - x^{(k)}) + (A^{(k+1)} - A^{(k)})(x - x^{(k)}) \\ &= (A^{(k+1)} - A^{(k)})(x - x^{(k)}) \end{aligned}$$

e se o vector $x - x^{(k)}$ for expresso como $\alpha(x^{(k+1)} - x^{(k)}) + t$ em que t é ortogonal a Δ_x, $t^T\Delta_x = 0$, tem-se

$$l(x) - l^{(k)}(x) = \alpha(A^{(k+1)} - A^{(k)})(x^{(k+1)} - x^{(k)}) + (A^{(k+1)} - A^{(k)})t.$$

Para reduzir $l(x) - l^{(k)}(x)$ e como o primeiro termo, de acordo com a condição secante, é igual a

$$\alpha(y^{(k)} - A^{(k)}\Delta_x),$$

só nos resta tentar anular o segundo termo. Assim $A^{(k+1)}$ deve ser escolhida de tal forma que

$$(A^{(k+1)} - A^{(k)})t = 0, \text{ sabendo que } t^T\Delta_x = 0.$$

A diferença $A^{(k+1)} - A^{(k)}$ deve ser uma matriz de característica igual a um com a forma $u\,\Delta_x^T$, $u \in \mathbf{R}^n$, isto é, uma matriz cujas linhas são vectores múltiplos do vector Δ_x, de tal forma que $u\,\Delta_x^T t = 0$.

Da condição secante

$$(A^{(k+1)} - A^{(k)})\Delta_x = y^{(k)} - A^{(k)}\Delta_x,$$

conclui-se que

$$u\,\Delta_x^T\Delta_x = y^{(k)} - A^{(k)}\Delta_x, \text{ ou } u = \frac{(y^{(k)} - A^{(k)}\Delta_x)}{\Delta_x^T\Delta_x}$$

o que origina a *menor variação no modelo linear*, consistente com (4.10),

$$A^{(k+1)} = A^{(k)} + \frac{(y^{(k)} - A^{(k)}\Delta_x)\Delta_x^T}{\Delta_x^T\Delta_x}. \tag{4.11}$$

A esta fórmula é costume chamar *fórmula de actualização de Broyden*. O termo actualização usa-se porque a matriz $A^{(k+1)}$, que é uma aproximação a $J(x^{(k+1)})$, surge a partir da actualização da matriz $A^{(k)}$ (aproximação a $J(x^{(k)})$).

O modelo linear (4.8) fica completo com a especificação da matriz $A^{(k+1)}$. O próximo elemento da sequência $\{x^{(k)}\}$, de aproximações à solução, é obtido como sendo a raiz deste modelo e isto equivale a substituir, nas equações Newton, $J(x^{(k)})$ por $A^{(k)}$. A equação iterativa do método da secante de Broyden é

$$A^{(k)}\Delta_x = -f(x^{(k)}),\ k = 1, 2, \ldots \tag{4.12}$$

em que

$$x^{(k+1)} = x^{(k)} + \Delta_x,$$
$$y^{(k)} = f(x^{(k+1)}) - f(x^{(k)})$$

e

$$A^{(k+1)} = A^{(k)} + \frac{(y^{(k)} - A^{(k)}\Delta_x)\Delta_x^T}{\Delta_x^T\Delta_x}.$$

O algoritmo correspondente a este processo iterativo encontra-se no algoritmo 4.3.

Para a primeira aproximação $A^{(1)}$ do processo iterativo, pode-se usar uma das seguintes alternativas: a matriz $J(x^{(1)})$ (que pressupõe que as derivadas foram calculadas analiticamente), a matriz que se obtém através da aproximação, das primeiras derivadas, pelas diferenças finitas, ou ainda a matriz identidade. O comportamento deste método vai depender, não só da matriz atribuída a $A^{(1)}$ mas também da primeira aproximação inicial, $x^{(1)}$.

Algoritmo 4.3 :

1. ler n e n_{max} e calcular tol
2. introduzir as funções $f_1(x_1, x_2, \ldots, x_n), f_2(x_1, x_2, \ldots, x_n), \ldots, f_n(x_1, x_2, \ldots, x_n)$
3. fazer $k = 0$

   3.1. para $j = 1, \ldots, n$ ler $x_j^{(1)}$

   3.2. para $i = 1, \ldots, n$ calcular $f_i(x_1^{(1)}, x_2^{(1)}, \ldots, x_n^{(1)})$

4. para $i = 1, \ldots, n$ e para $j = i, \ldots, n$

   4.1. se $j = i$ então fazer $a_{ij}^{(1)} = 1$

4.2. senão fazer $a_{ij}^{(1)} = a_{ji}^{(1)} = 0$

5. fazer $k = k + 1$

6. resolver o sistema linear (algoritmo 3.3), em ordem ao vector Δ_x:

$$\begin{pmatrix} a_{11}^{(k)} & a_{12}^{(k)} & \dots & a_{1n}^{(k)} \\ a_{21}^{(k)} & a_{22}^{(k)} & \dots & a_{2n}^{(k)} \\ . & . & & . \\ . & . & & . \\ . & . & & . \\ a_{n1}^{(k)} & a_{n2}^{(k)} & \dots & a_{nn}^{(k)} \end{pmatrix} \Delta_x = - \begin{pmatrix} f_1(x^{(k)}) \\ f_2(x^{(k)}) \\ . \\ . \\ . \\ f_n(x^{(k)}) \end{pmatrix}$$

7. para $j = 1, \dots, n$ calcular $x_j^{(k+1)} = x_j^{(k)} + \Delta_{x_j}$

8. para $i = 1, \dots, n$ calcular $f_i(x_1^{(k+1)}, x_2^{(k+1)}, \dots, x_n^{(k+1)})$

9. se convergência=.TRUE. (algoritmo 4.1) então ir para o passo 12

10. se $k \geq n_{max}$ então recomeçar, ir para o passo 3

11. actualizar a matriz A:

11.1. para $i = 1, \dots, n$ calcular $y_i = f_i(x^{(k+1)}) - f_i(x^{(k)})$

11.2. para $i = 1, \dots, n$ calcular $aux_i = y_i - \sum_{j=1}^{n} a_{ij} \Delta_{x_j}$

11.3. calcular $den = \sum_{i=1}^{n} (\Delta_{x_i})^2$

11.4. se $den \leq tol$ então terminar com a última aproximação

11.5. para $i = 1, \dots, n$ e $j = 1, \dots, n$ calcular $a_{ij}^{(k+1)} = a_{ij}^{(k)} + \frac{aux_i \Delta_{x_j}}{den}$

11.6. ir para o passo 5

12. terminar com $x^* \leftarrow (x_j^{(k+1)}, j = 1, \dots, n)$.

4.3.2 Convergência do método de Broyden

Vamos analisar a convergência local do método de Broyden. Veremos que, se a aproximação inicial, $x^{(1)}$, estiver suficientemente perto da solução x^* e se $A^{(1)}$ também estiver perto de $J(x^*)$, com $J(x^*)$ uma matriz não singular, então a sequência de aproximações $\{x^{(k)}\}$ converge superlinearmente para x^*. Considere-se o seguinte lema.

Lema 4.3 : Seja $D \subseteq \mathbb{R}^n$ um conjunto aberto e convexo que contém $x^{(k)}$ e $x^{(k+1)}$, com $x^{(k)} \neq x^{(k+1)}$. Sejam ainda $f : \mathbb{R}^n \to \mathbb{R}^n$, a matriz $J(x)$ Lipschitz contínua no conjunto D, com constante γ, $A^{(k)} \in \mathbb{R}^{n \times n}$ e a matriz $A^{(k+1)}$ definida por (4.11). Então, quer para a norma Frobenius quer para a norma 2 de matrizes, tem-se

$$\|A^{(k+1)} - J(x^{(k+1)})\| \leq \|A^{(k)} - J(x^{(k)})\| + \frac{3\gamma}{2} \|x^{(k+1)} - x^{(k)}\|_2. \tag{4.13}$$

Além disso, se $x^* \in D$ *e* $J(x)$ *verifica a condição de Lipschitz mais fraca*

$$\|J(x) - J(x^*)\| \leq \gamma \|x - x^*\|$$

então

$$\|A^{(k+1)} - J(x^*)\| \leq \|A^{(k)} - J(x^*)\| + \frac{\gamma}{2}(\|x^{(k+1)} - x^*\|_2 + \|x^{(k)} - x^*\|_2). \tag{4.14}$$

As desigualdades (4.13) e (4.14) caracterizam a *propriedade da deterioração limitada* e significam que se a aproximação ao Jacobiano piora, de iteração para iteração, isso verifica-se de uma maneira controlada e limitada. É possível provar que se as matrizes A satisfazem (4.14) então a sequência $\{x^{(k)}\}$ converge pelo menos linearmente para x^*.

Teorema 4.4 : Seja $f : \mathbb{R}^n \to \mathbb{R}^n$ *uma função continuamente diferenciável num conjunto aberto e convexo* $D \subset \mathbb{R}^n$. *Suponha que existem* $x^* \in \mathbb{R}^n$ *e* $r, \beta > 0$, *tais que* $V(x^*, r) \subset D$, $f(x^*) = 0$, *a matriz* $J(x^*)^{-1}$ *existe,* $\|J(x^*)^{-1}\| \leq \beta$ *e* $J(x)$ *é Lipschitz contínua no conjunto* $V(x^*, r)$, *com constante* γ. *Suponha ainda que existem constantes positivas* ε *e* δ *tais que se*

$$\|x^{(1)} - x^*\|_2 \leq \varepsilon$$

e

$$\|A^{(1)} - J(x^*)\| \leq \delta,$$

então a sequência $\{x^{(k)}\}$ *gerada pelo método de Broyden é bem definida e converge superlinearmente para* x^*.

Se a sequência $\{A^{(k)}\}$ *apenas verifica a propriedade (4.14), então* $\{x^{(k)}\}$ *converge pelo menos linearmente para* x^*.

A propriedade da deterioração limitada (4.14) assegura que $\|A^{(k+1)} - J(x^*)\|$ se conserva razoavelmente pequena. Se as aproximações ao Jacobiano apenas satisfazem (4.14), então não se consegue mais do que uma convergência linear. Isto significa que o processo iterativo converge, mesmo que a aproximação ao Jacobiano não seja muito boa. No entanto, o que nos interessa é ter uma convergência superlinear. Esta consegue-se localmente de acordo com as condições enunciadas no Teorema 4.4.

Uma condição necessária e suficiente para que um método do tipo secante, e nomeadamente o método de Broyden, convirja superlinearmente para x^*, é a de que os deslocamentos efectuados pelo método da secante convirjam, quer em direcção quer em amplitude, para os deslocamentos Newton (veja-se equação (4.3)) quando efectuados a partir dos mesmos pontos. O teorema seguinte estabelece isto mesmo.

Teorema 4.5 : Seja $D \subseteq \mathbb{R}^n$ *um conjunto aberto e convexo e sejam ainda* $f : \mathbb{R}^n \to \mathbb{R}^n$, $J(x)$ *uma matriz Lipschitz contínua em* D, *com constante* γ, $x^* \in D$ *e a matriz* $J(x^*)$ *não singular. Seja* $\{A^{(k)}\}$ *uma sequência de matrizes não singulares em* $\mathbb{R}^{n \times n}$ *e suponha que para um* $x^{(1)} \in D$ *a sequência de pontos gerada por*

$$x^{(k+1)} = x^{(k)} + \Delta_x$$

com

$$A^{(k)}\Delta_x = -f(x^{(k)})$$

permanece no conjunto D *e satisfaz* $x^{(k)} \neq x^*$, *para qualquer* k, *e*

$$lim_{k\to\infty} x^{(k)} = x^*.$$

Então a sequência $\{x^{(k)}\}$ *converge superlinearmente para* x^*, *de acordo com alguma norma* $\|.\|$ *e* $f(x^*) = 0$ *se e só se*

$$lim_{k\to\infty} \frac{\|(A^{(k)} - J(x^*))(x^{(k+1)} - x^{(k)})\|}{\|x^{(k+1)} - x^{(k)}\|} = 0. \tag{4.15}$$

De acordo com a condição (4.15), quando o método de Broyden converge superlinearmente para uma raiz x^*, não é possível concluir que a matriz $A^{(k)}$, que aproxima o Jacobiano, esteja a convergir para $J(x^*)$, embora possa acontecer.

4.3.3 Implementação. Rotina BROYDE

A implementação do algoritmo 4.3 origina a rotina BROYDE. Os resultados obtidos com a rotina são os seguintes:

Exemplo 4.5 Resolvendo o sistema do exemplo 4.1 obtém-se a solução $(1.086187, 1.943685)^T$ ao fim de 9 iterações, usando o mesmo vector inicial $(1, 2)^T$. Em geral, o método da secante é mais lento do que o método de Newton, como se verifica neste caso.

Exemplo 4.6 A partir do sistema do exemplo 4.2 e do vector inicial $(0, 0)^T$ (desta vez a matriz do sistema não é singular uma vez que se trabalha com uma aproximação ao Jacobiano) obtém-se a solução $(2.561553, -1.561553)^T$ ao fim de 10 iterações. Este ponto de intersecção das duas curvas foi também encontrado no exemplo 4.2.

Exemplo 4.7 A partir do vector inicial $(2, 3)^T$, o processo não converge para a solução do sistema do exemplo 4.3. Atribuindo outro valor inicial, o vector $(2, 0.9)^T$, ao fim de 10 iterações consegue-se a solução $(2, 1)^T$. Quatro das iterações obtidas no processo são:

k	$x^{(k+1)}$	k	$x^{(k+1)}$
1	$(1.929000, 1.400000)^T$	4	$(2.004273, 0.970609)^T$
2	$(2.163696, 1.037979)^T$	7	$(2.000532, 0.999592)^T$

4.4 Problemas

1. O sistema das equações não lineares

$$\begin{cases} x_1^2 + (x_2 - 2)^2 = 9 \\ (x_1 - 1)^2 + x_2^2 = 2 \end{cases}$$

tem uma solução próxima de $(-0.1, -1)^T$. Calcule-a usando o método iterativo de Newton.

Pare o processo iterativo quando as condições do critério de paragem forem verificadas para $\varepsilon_1 = 0.1$ e $\varepsilon_2 = 0.01$.

2. O sistema

$$\begin{cases} -5x_1 + 2sen(x_1) + cos(x_2) &= 0 \\ 4cos(x_1) + 2sen(x_2) - 5x_2 &= 0 \end{cases}$$

tem uma única solução. Calcule-a

   a) Usando o método de Newton. Considere os seguintes parâmetros de entrada: $x^{(1)} = (0, 0)^T$ e $\varepsilon_1 = \varepsilon_2 = 10^{-6}$ (no critério de paragem).

   b) Usando o método da secante baseado na equação Broyden. Use os mesmos parâmetros de entrada. Verifique que a solução $(0.132976, 1.159594)^T$ é calculada ao fim de duas vezes mais iterações do que as necessárias em a).

3. Considere a equação $f(z) = z^3 - 2z - 5 = 0$, em que $z = x_1 + x_2 i$. Calcule uma das raízes, com uma precisão de 10^{-4}, aplicando o método de Newton ao sistema equivalente de duas equações não lineares. Compare estes resultados com os obtidos na resolução do problema 10 do Capítulo 2. Tome para vector inicial $(-1, 1)^T$.

4. O sistema

$$\begin{cases} 3x_1 - cos(x_2 x_3) &= 0.5 \\ x_1^2 - 625x_2^2 &= 0 \\ e^{-x_1 x_2} + 20x_3 &= -9 \end{cases}$$

tem mais do que uma solução.

   a) Calcule o Jacobiano. Use o método de Newton para calcular uma das soluções. O que é que acontece quando o vector inicial é $(0, 0, 0)^T$? E se tomar $x^{(1)} = (1, 0, 1)^T$? Repita o processo iterativo considerando agora $x^{(1)} = (4, 0.5, -0.5)^T$. Quantas iterações calculou até obter aproximadamente o vector

$$(0.499983, 0.019999, -0.499503)^T.$$

   b) Use o método da secante de Broyden até atingir aproximadamente o vector

$$(0.499983, -0.019999, -0.500502)^T.$$

   Tome para aproximação inicial $(0.5, 0, -0.5)^T$.

   c) O que aconteceria ao método da secante se iniciasse o processo iterativo com o vector $(1, 0, 1)^T$?

Capítulo 5

Valores e vectores próprios de matrizes

5.1 Introdução

5.1.1 Forma geral do problema

Dada uma matriz real, A, quadrada de ordem n, os seus valores próprios e os vectores que lhes estão associados estão relacionados através da seguinte *equação do valor próprio*

$$Ax = \lambda x\,, \tag{5.1}$$

em que λ é um escalar, conhecido por *valor próprio* de A e x é um vector de n elementos, conhecido por *vector próprio* de A associado a λ. Uma matriz de ordem n tem n valores próprios $\lambda_1, \lambda_2, \ldots, \lambda_n$ e n vectores próprios associados $x_1, x_2, \ldots, x_n$. Os valores próprios de uma matriz A real podem ser reais e/ou complexos. Se tiver valores complexos estes surgem aos pares conjugados e os correspondentes vectores são também complexos conjugados.

5.1.2 Características do problema

Os valores próprios da matriz A e da sua transposta A^T são iguais e verificam

$$Ax_i = \lambda_i x_i \quad A^T y_i = \lambda_i y_i \quad \text{para} \quad i = 1, \ldots, n.$$

Se os vectores próprios x_i da matriz A e y_i da matriz A^T, $i = 1, \ldots, n$ estiverem normalizados de acordo com a norma 2,

$$\|x_i\|_2 = x_i^T x_i = 1, \quad \|y_i\|_2 = y_i^T y_i = 1,$$

e se a matriz for simétrica, $A = A^T$, então verifica-se

$$x_i^T y_i = 1, \quad i = 1, \ldots, n.$$

Se a matriz A é real, do tipo $n \times n$ e simétrica, $A = A^T$, então os seus valores próprios $\lambda_1, \lambda_2, \ldots, \lambda_n$ são reais e os n vectores próprios associados são independentes e reais. Se a matriz A é simétrica e definida positiva, os seus valores próprios são todos reais e positivos.

5.1.3 Tipos de métodos. Critério de paragem

A equação (5.1) pode ser escrita na forma

$$(A - \lambda I)x = 0,$$

que para um valor fixo de λ se reduz a um sistema de equações lineares homogéneo, cuja matriz dos coeficientes é $A - \lambda I$. Uma solução trivial do sistema é $x = 0$. Para se obter uma solução diferente do vector nulo, algum elemento diagonal da matriz U, triangular superior da condensação de $A - \lambda I$, terá de ser nulo. Isto significa que o determinante de U ou o determinante de $A - \lambda I$ deverá ser igual a zero. A equação

$$det(A - \lambda I) = 0$$

chama-se *equação característica* e as suas soluções são os valores próprios de A, $\lambda_1, \ldots, \lambda_n$. Esta equação é definida por um polinómio de grau n em λ.

Para calcular os valores próprios de uma matriz, não é aconselhável calcular as raízes da equação característica. Os coeficientes do polinómio são obtidos, apenas, aproximadamente e, os zeros do polinómio, especialmente os múltiplos, são muito sensíveis a pequenas perturbações nos coeficientes.

Assim, aplica-se uma transformação não singular à equação do valor próprio com o objectivo de a simplificar e facilitar o cálculo dos valores e vectores próprios.

Considere as matrizes X e D definidas da seguinte maneira:

$$X = [x_1 \ x_2 \ \ldots \ x_n]$$

e

$$D = diag(\lambda_i)\,, \ i = 1, 2, \ldots, n$$

ou seja, as colunas da matriz X são os vectores próprios de A, x_i, $i = 1, \ldots, n$ e a matriz D é diagonal, em que os elementos diferentes de zero são os valores próprios de A. Estas matrizes satisfazem

$$AX = XD \tag{5.2}$$

e como X é uma matriz não singular, pode-se colocar esta equação na forma

$$X^{-1}AX = D,$$

que caracteriza uma *transformação semelhante* especial. Falaremos, mais adiante de outras transformações semelhantes,

$$Q^{-1}AQ = B,$$

em que Q é uma matriz não singular. Como $A = QBQ^{-1}$, também

$$AX = XD \text{ se transforma em } QBQ^{-1}X = XD$$

ou seja

$$B(Q^{-1}X) = (Q^{-1}X)D. \tag{5.3}$$

Das equações (5.2) e (5.3) pode-se concluir que

- os valores próprios das matrizes A e B são iguais;
- se x_i é um vector próprio da matriz A, então $Q^{-1}x_i$ é um vector próprio de B.

Em muitas aplicações, interessa calcular, não o conjunto inteiro dos valores próprios de uma matriz, mas apenas alguns, designadamente o valor próprio de maior módulo ou o de menor módulo. Para estes casos particulares torna-se vantajoso usar um método simples. O *método da potência*, também conhecido por iteração de Von Mises[1] consiste num processo iterativo de fácil implementação e que serve para calcular o valor próprio de maior módulo e o vector associado, se a implementação for directa, baseada na matriz A, ou o valor próprio de menor módulo e o vector associado, se a implementação for baseada na matriz inversa, A^{-1}. Por se tratar de um processo iterativo, o número de iterações efectuadas tem de ser finito. Para se conhecer a altura certa para a paragem do processo, é preciso verificar a proximidade do valor próprio calculado em relação ao valor exacto. Faz-se o mesmo em relação ao vector associado. O algoritmo 5.1 descreve o critério de paragem, que deve ser usado com o método da potência. É baseado nas duas condições

$$\frac{|\lambda^{(k+1)} - \lambda^{(k)}|}{|\lambda^{(k+1)}|} \leq \varepsilon_1$$

e

$$\frac{\|x^{(k+1)} - x^{(k)}\|}{\|x^{(k+1)}\|} \leq \varepsilon_2$$

sendo ε_1 e ε_2 quantidades pequenas e positivas, $\lambda^{(k)}$ o valor próprio e $x^{(k)}$ o vector próprio associado calculados na iteração k.

Algoritmo 5.1 :

1. `ler` n, k, $\lambda^{(k)}$, $\lambda^{(k+1)}$, ε_1, ε_2 `e para` $i = 1, \ldots, n$ `ler` $x_i^{(k)}$, $x_i^{(k+1)}$ `e calcular` tol
2. `calcular` $abs1 = |\lambda^{(k+1)} - \lambda^{(k)}|$
3. `calcular` $abs2 = |\lambda^{(k+1)}|$
4. `calcular` $norma1 = \sqrt{\sum_{i=1}^{n}(x_i^{(k+1)} - x_i^{(k)})^2}$

[1] R. E. Von Mises viveu entre 1883 e 1953. Trabalhou em Viena, Berlim, Istambul e na Universidade de Harvard nos Estados Unidos da América e além do método iterativo da potência para calcular valores e vectores próprios, Von Mises desenvolveu trabalho de investigação na resolução de sistemas de equações lineares, no cálculo das probabilidades e na teoria da camada limite (Hämmerlin e Hoffmann (1991)).

5. `calcular` $norma2 = \sqrt{\sum_{i=1}^{n} (x_i^{(k+1)})^2}$
6. `se` $abs2 \leq tol$ `então` $cri1 = abs1$ `senão` $cri1 = \frac{abs1}{abs2}$
7. `se` $norma2 \leq tol$ `então` $cri2 = norma1$ `senão` $cri2 = \frac{norma1}{norma2}$
9. `se` $cri1 \leq \varepsilon_1$ `e` $cri2 \leq \varepsilon_2$ `então terminar com convergência=.TRUE.`
10. `terminar com convergência=.FALSE.`

5.1.4 Índice de algoritmos

A listagem dos dez algoritmos que são apresentados ao longo deste Capítulo é a que se segue.

Algoritmo 5.1 Critério de paragem para o método da potência

Algoritmo 5.2 Cálculo do valor próprio de maior módulo e correspondente vector próprio, pelo método da potência, iteração directa

Algoritmo 5.3 Cálculo do valor próprio de menor módulo e correspondente vector próprio, pelo método da potência, iteração inversa

Algoritmo 5.4 Cálculo do valor próprio mais próximo de uma estimativa e correspondente vector próprio pelo método da potência, iteração inversa e deslocada

Algoritmo 5.5 Transformação de uma matriz simétrica à forma tridiagonal simétrica pelo método de Givens

Algoritmo 5.6 Transformação de uma matriz simétrica à forma tridiagonal simétrica pelas matrizes de Householder

Algoritmo 5.7 Cálculo dos valores próprios de uma matriz tridiagonal simétrica pelas sequências de Stürm e método de Newton

Algoritmo 5.8 Transformação de uma matriz não simétrica à forma quase triangular superior por transformações elementares

Algoritmo 5.9 Cálculo dos valores próprios de uma matriz na forma quase triangular superior pelo método Q-R

Algoritmo 5.10 Cálculo do vector próprio que corresponde a um valor próprio complexo pelo método da potência, iteração inversa e deslocada

A escolha deve ser feita tendo como base os valores e vectores próprios que se querem calcular. Se for necessário calcular apenas uma solução individual (por exemplo, o valor próprio de maior módulo e o vector que lhe está associado) os algoritmos 5.2, 5.3 ou 5.4 devem ser os seleccionados. Se interessar calcular os n valores próprios e se a matriz for simétrica então a escolha deve cair sobre o algoritmo 5.5 seguido do 5.7, ou, primeiro o 5.6 seguido do 5.7. Se a matriz dada não for simétrica então os valores próprios podem ser calculados usando o algoritmo 5.8 seguido do 5.9. Para calcular vectores próprios associados a valores reais, o algoritmo 5.4 deve ser implementado para cada valor e se os valores próprios forem complexos então o algoritmo 5.10 calcula o vector associado.

5.2 Condicionamento do problema

5.2.1 Condicionamento de um valor próprio

Analisemos o condicionamento do valor próprio λ_i, para qualquer i. A introdução de pequenas perturbações nos dados do problema, matriz A, pode originar variações significativas nos valores próprios, concluindo-se que o respectivo valor é *mal condicionado*. Sejam λ_i o valor próprio de A e $\lambda_i(\epsilon)$ o valor próprio calculado a partir da matriz perturbada $A+\epsilon B$. Considere ϵ positivo e próximo de zero e $|b_{jk}|<1$, sendo $B=(b_{jk}), (j,k=1,\ldots,n)$. Se x_i e $x_i(\epsilon)$ forem os vectores próprios de A e $A+\epsilon B$ associados respectivamente a λ_i e $\lambda_i(\epsilon)$, tem-se que

$$Ax_i=\lambda_i x_i \quad \text{e} \quad (A+\epsilon B)x_i(\epsilon)=\lambda_i(\epsilon)x_i(\epsilon) \tag{5.4}$$

com

$$\lambda_i(\epsilon)=\lambda_i+k_i\epsilon+\mathcal{O}(\epsilon^2) \tag{5.5}$$

e

$$x_i(\epsilon)=x_i+\epsilon(t_{11}x_1+\cdots+t_{i-1\,1}x_{i-1}+t_{i+1\,1}x_{i+1}+\cdots+t_{n\,1}x_n)+\mathcal{O}(\epsilon^2). \tag{5.6}$$

Substituindo (5.5) e (5.6) na segunda equação de (5.4), usando $Ax_i=\lambda_i x_i$ e ignorando os termos da ordem de ϵ^2, fica-se com a igualdade

$$\sum_{k=1,k\neq i}^{n}(\lambda_k-\lambda_i)t_{k1}x_k+Bx_i=k_ix_i. \tag{5.7}$$

Multiplicando esta igualdade, à esquerda, por y_i^T, vector próprio de A^T que corresponde ao valor λ_i, tem-se

$$y_i^T\sum_{k=1,k\neq i}^{n}(\lambda_k-\lambda_i)t_{k1}x_k+y_i^TBx_i=k_iy_i^Tx_i$$

e como $y_i^Tx_k=0$, para $k\neq i$, obtém-se para a perturbação em λ_i,

$$k_i=\frac{y_i^TBx_i}{y_i^Tx_i} \ \text{ e } \ |k_i|\leq\frac{n}{|s_i|} \tag{5.8}$$

em que $s_i=y_i^Tx_i$ e

$$|y_i^TBx_i|\leq\|y_i^T\|_2\|B\|_2\|x_i\|_2\leq\|B\|_2\leq n,$$

uma vez que os vectores estão normalizados, isto é $\|x_i\|_2=\|y_i\|_2=1, i=1,\ldots,n$. Daqui se conclui, que se $|s_i|\approx 0$, k_i atinge valores arbitrariamente grandes, $\lambda_i(\epsilon)$ está muito afastado de λ_i (veja-se (5.5)) e nesta situação o valor próprio diz-se *mal condicionado*. A quantidade $\frac{1}{|s_i|}$ é considerada como *número de condição* uma vez que determina a sensibilidade do valor próprio relativamente a variações nos dados. Quando a matriz A é simétrica, $s_i=1$ para qualquer i, o número de condição nunca atinge valores enormes e os valores próprios são sempre bem condicionados.

5.2.2 Condicionamento de um vector próprio

Para se analisar a sensibilidade do vector próprio x_i, que corresponde ao valor λ_i, para qualquer i, deve-se verificar os efeitos, que pequenas perturbações em A, originam no vector. As equações (5.4), (5.5) e (5.6) continuam a ser válidas. Usando a igualdade (5.7) e multiplicando, à esquerda, por y_r^T, com $r \neq i$, tem-se

$$y_r^T \sum_{k=1,k\neq i}^{n} (\lambda_k - \lambda_i) t_{k1} x_k + y_r^T B x_i = k_i y_r^T x_i$$

ou

$$(\lambda_r - \lambda_i) t_{r1} y_r^T x_r + y_r^T B x_i = 0,$$

donde se retira o valor de t_{r1},

$$t_{r1} = -\frac{y_r^T B x_i}{(\lambda_r - \lambda_i) s_r} \tag{5.9}$$

em que $s_r = y_r^T x_r$, para todo o r.

Da equação (5.6) tira-se que a perturbação de primeira ordem (da ordem de ϵ) sofrida pelo vector x_i é

$$x_i(\epsilon) - x_i = \epsilon(t_{11}x_1 + \cdots + t_{i-1\,1}x_{i-1} + t_{i+1\,1}x_{i+1} + \cdots + t_{n1}x_n)$$

ou, usando (5.9),

$$\|x_i(\epsilon) - x_i\| \leq \epsilon \sum_{k=1,k\neq i}^{n} \frac{n}{|\lambda_k - \lambda_i||s_k|} \tag{5.10}$$

uma vez que $|y_k^T B x_i| \leq \|y_k^T\|_2 \|B\|_2 \|x_i\|_2 \leq n$ e $\|x_k\|_2 = 1$ para todo o k.

Assim, esta perturbação pode atingir valores enormes devido a dois factores. O primeiro está relacionado com a diferença $|\lambda_k - \lambda_i|$. Se ela for próxima de zero, o que equivale a ter um valor próprio muito próximo de λ_i (valor próprio associado ao vector cujo condicionamento se está a analisar) então o vector x_i pode ser *mal condicionado*. Esta análise é válida tanto para matrizes simétricas como para as não simétricas. O outro factor é a quantidade $|s_k|$. Se ela atingir valores muito próximos de zero, o vector próprio x_i pode sofrer uma perturbação enorme e ser *mal condicionado*. Tal como foi referido em 5.2.1 para matrizes simétricas, o factor s_k não influencia a condição, uma vez que é sempre igual a um.

5.3 Método iterativo da potência

5.3.1 Valor próprio de maior módulo. Convergência

Suponha que os valores próprios de A estão ordenados de tal modo que

$$|\lambda_1| > |\lambda_2| > \ldots > |\lambda_n|$$

isto é, o valor próprio de maior módulo é o λ_1 e o de menor módulo é o λ_n. Um processo iterativo muito simples que calcula o valor próprio de maior módulo, λ_1, e o vector próprio associado, x_1, ou o valor de menor módulo, λ_n, e o correspondente vector x_n, conhecidos por *soluções individuais*, é o *método da potência*. A equação iterativa será *directa* se o objectivo é calcular o valor próprio de maior módulo, será *inversa* se interessar calcular o valor de menor módulo (veja-se em 5.3.3) e será *inversa e deslocada* se o que interessa é calcular o valor próprio que se encontra mais próximo de uma estimativa dada. Este último tipo de equação iterativa é vantajoso no cálculo do vector próprio da matriz A que corresponde a um certo valor próprio já conhecido (veja-se em 5.3.5).

Para o cálculo do valor λ_1 e do correspondente x_1, de A, a equação iterativa do *método da potência, iteração directa*, é definida por

$$w^{(k+1)} = Av^{(k)}, \quad k = 1, 2, \ldots \tag{5.11}$$

com $v^{(1)}$ vector arbitrário, mas normalizado de acordo com a norma infinita, e

$$v^{(k)} = \frac{w^{(k)}}{|w^{(k)}|_\infty}$$

sendo a quantidade $|w^{(k)}|_\infty$ definida como *o elemento do vector $w^{(k)}$ que tem maior módulo*. Deste modo, o vector resultante $v^{(k)}$ fica normalizado de acordo com a norma infinita, isto é, $\|v^{(k)}\|_\infty = 1$.

Quando $k \to \infty$, o vector $w^{(k)}$ tende para um múltiplo de x_1 e $v^{(k)} \to x_1$. Durante este processo iterativo, o valor próprio λ_1 também é calculado. Verifica-se que, quando $k \to \infty$, o valor $|w^{(k)}|_\infty$, calculado ao longo das iterações, tende para λ_1. O algoritmo que corresponde a este processo iterativo é o algoritmo 5.2.

Algoritmo 5.2 :

1. ler n, n_{max} e para $i = 1, \ldots, n$ ler $(a_{ij}, j = 1, \ldots, n)$
2. para $i = 1, \ldots, n$ fazer $x_i^{(1)} = v_i^{(1)} = 1$, $\lambda^{(1)} = 1$ (ou outros) e fazer $k = 0$
3. fazer $k = k + 1$
4. para $i = 1, \ldots, n$ calcular $w_i^{(k+1)} = \sum_{j=1}^{n} a_{ij} v_j^{(k)}$
5. para $i = 1, \ldots, n$ calcular $elem$ tal que $|w_{elem}^{(k+1)}| \geq |w_i^{(k+1)}|$
6. para $i = 1, \ldots, n$ calcular $v_i^{(k+1)} = \frac{w_i^{(k+1)}}{w_{elem}^{(k+1)}}$
7. fazer $\lambda^{(k+1)} = w_{elem}^{(k+1)}$ e para $i = 1, \ldots, n$ fazer $x_i^{(k+1)} = v_i^{(k+1)}$
8. se convergência=.FALSE. (algoritmo 5.1) então
   - 8.1. se $k \geq n_{max}$ então recomeçar, ir para o passo 2
   - 8.2. ir para o passo 3
9. terminar com $\lambda_1 \leftarrow \lambda^{(k+1)}$ e $x_1 \leftarrow (v_i^{(k+1)}, i = 1, \ldots, n)$.

A convergência desta iteração directa do método da potência depende das quantidades $\frac{\lambda_i}{\lambda_1}$, $i = 2, \ldots, n$. Como o vector $v^{(1)}$ se pode exprimir como uma combinação linear dos x_i, $i = 1, \ldots, n$, vectores próprios de A linearmente independentes,

$$v^{(1)} = \sum_{i=1}^{n} \alpha_i x_i$$

e como de (5.11)

$$w^{(k+1)} = A^k v^{(1)},$$

tem-se[2]

$$w^{(k+1)} = \sum_{i=1}^{n} \alpha_i \lambda_i^k x_i.$$

Esta igualdade pode ser escrita na forma

$$w^{(k+1)} = \lambda_1^k (\alpha_1 x_1 + \sum_{i=2}^{n} \alpha_i (\frac{\lambda_i}{\lambda_1})^k x_i) \tag{5.12}$$

donde se conclui que quanto menores forem os quocientes $\frac{\lambda_i}{\lambda_1}$ (são sempre menores do que um) mais insignificante é a contribuição do segundo termo de (5.12) na formação de $w^{(k+1)}$ e mais rápida é a convergência de $w^{(k+1)}$ para um múltiplo de x_1. Esta condição exige um afastamento dos valores próprios $\lambda_2, \ldots, \lambda_n$ em relação ao λ_1.

Quando o valor próprio de maior módulo é um número complexo e como estes aparecem sempre aos pares conjugados, têm-se dois valores próprios com o mesmo módulo

$$|l_1 + l_2 i| = |l_1 - l_2 i| = \sqrt{l_1^2 + l_2^2},$$

donde se conclui que a iteração directa do método da potência não vai convergir para o valor de maior módulo, uma vez que existem dois e o processo converge alternadamente para um e para o outro.

5.3.2 Implementação. Rotina MAIVAL

A rotina MAIVAL implementa o algoritmo 5.2. Foram usados os valores $\varepsilon_1 = \varepsilon_2 = 10^{-6}$ como tolerâncias para o critério de paragem, definido no algoritmo 5.1.

[2] Teorema: Dada uma matriz A quadrada de ordem n, tem-se que:

a) se λ é um valor próprio de A então λ^k é um valor próprio de A^k;

b) λ é um valor próprio de A não singular, se e só se λ^{-1} é um valor próprio de A^{-1};

c) as matrizes A e A^T têm os mesmos valores próprios;

d) se as duas matrizes A e B são quadradas de ordem n, então as matrizes AB e BA têm os mesmos valores próprios.

Exemplo 5.1 A matriz A

$$\begin{pmatrix} 1 & 1 & 0 \\ -0.99999999 & 3 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

tem dois valores próprios iguais a 2 e um igual a 0. A rotina MAIVAL mesmo assim consegue calcular $\lambda_1 = 2.000000015$ e $x_1 = (0.999999990, 1, 0.499999996)^T$ ao fim de 2 iterações.

Exemplo 5.2 O valor próprio de maior módulo de

$$A = \begin{pmatrix} 2 & 6 \\ -2 & -5 \end{pmatrix}$$

calculado pela rotina MAIVAL é -2.000001, sendo o vector associado $x_1 = (1, -0.666667)^T$. O processo iterativo leva 21 iterações.

Exemplo 5.3 A matriz

$$A = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$$

tem dois valores iguais a um. O processo iterativo directo é muito lento. Os valores obtidos ao longo das iterações são:

k	$\lambda^{(k+1)}$	$v^{(k+1)}$	k	$\lambda^{(k+1)}$	$v^{(k+1)}$
1	2	$(1, 0.5)^T$	25	1.04	$(1, 0.038462)^T$
5	1.2	$(1, 0.166667)^T$	35	1.028571	$(1, 0.027778)^T$
10	1.1	$(1, 0.90909)^T$	45	1.022222	$(1, 0.021739)^T$
15	1.066667	$(1, 0.0625)^T$	50	1.02	$(1, 0.019608)^T$

Exemplo 5.4 A matriz

$$A = \begin{pmatrix} 4 & -1 \\ 2 & 2 \end{pmatrix}$$

tem dois valores próprios complexos e conjugados $3 + \imath$ e $3 - \imath$. Tal como se esperava, a rotina MAIVAL não converge. A tabela seguinte mostra algumas iterações:

k	$\lambda^{(k+1)}$	k	$\lambda^{(k+1)}$	k	$\lambda^{(k+1)}$
1	4	17	-4.482798	35	2.106265
5	2.451613	25	2.234369	36	-2.495479
7	-3.410959	26	-1.951071	37	4.501812
15	2.348375	27	4.781347	45	1.959475

5.3.3 Valor próprio de menor módulo. Convergência

O método da potência também pode ser usado para calcular o valor próprio de menor módulo e o vector associado, de uma matriz A. Se $\lambda_1, \ldots, \lambda_n$ forem os valores próprios de A então $\frac{1}{\lambda_1}, \ldots, \frac{1}{\lambda_n}$ são valores próprios de A^{-1}. Se pensarmos em λ_n, valor próprio de

menor módulo de A, e em $\frac{1}{\lambda_n}$, valor próprio de maior módulo de A^{-1}, pode-se obter λ_n através de $\frac{1}{\lambda_n}$ usando o método da potência baseado em A^{-1}. Por este facto, a iteração se chama *inversa*. Assim, a equação iterativa é definida por

$$w^{(k+1)} = A^{-1}v^{(k)}, \text{ para } k = 1, 2, \ldots \tag{5.13}$$

com $v^{(1)}$ vector arbitrário e

$$v^{(k)} = \frac{w^{(k)}}{|w^{(k)}|_\infty}$$

sendo o valor $|w^{(k)}|_\infty$ definido como *o elemento do vector $w^{(k)}$ que tem maior módulo*. Deste modo, o vector $v^{(k)}$ fica normalizado de acordo com a norma infinita, isto é, $\|v^{(k)}\|_\infty = 1$. À medida que $k \to \infty$, o vector $w^{(k)}$ tende para um múltiplo de x_n, vector próprio de A associado a λ_n. Também se tem

$$v^{(k)} \to x_n \text{ quando } k \to \infty,$$

e o valor de λ_n determina-se a partir de

$$|w^{(k)}|_\infty \to \frac{1}{\lambda_n} \text{ quando } k \to \infty.$$

A *iteração inversa do método da potência* encontra-se no algoritmo 5.3.

Para que não seja necessário calcular a inversa da matriz A, a equação iterativa (5.13) pode ser reformulada da seguinte maneira

$$Aw^{(k+1)} = v^{(k)}, \text{ para } k = 1, 2, \ldots \tag{5.14}$$

o que equivale a resolver um sistema de equações lineares, em todas as iterações, com matriz dos coeficientes, A, constante e vector do termo independente, $v^{(k)}$, que varia de iteração para iteração. A solução $w^{(k+1)}$ é diferente em cada iteração. A matriz A é condensada à forma U, usando a eliminação de Gauss com pivotagem parcial e o sistema (5.14) é equivalente a

$$Uw^{(k+1)} = Jv^{(k)}, \text{ para } k = 1, 2, \ldots \tag{5.15}$$

em que a matriz J faz parte do processo de condensação de A, $JA = U$ (veja-se em 3.5.1 no Capítulo 3).

Algoritmo 5.3 :

1. `ler` n, n_{max}, `calcular` tol `e para` $i = 1, \ldots, n$ `ler` $(a_{il}, l = 1, \ldots, n)$
2. `calcular os factores J e U da matriz` A, `tais que` $JA = U$ `(algoritmo 3.5, com` *selec*`="J e U")`
3. `para` $i = 1, \ldots, n$ `fazer` $x_i^{(1)} = v_i^{(1)} = 1$, $\lambda^{(1)} = 1$ `(ou outros) e fazer` $k = 0$
4. `fazer` $k = k + 1$
5. `resolver o sistema` $Uw^{(k+1)} = Jv^{(k)}$ `em ordem a` $w^{(k+1)}$:

5.1. para $i = 1, \ldots, n$ calcular $aux_i = \sum_{l=1}^{n} j_{il} v_l^{(k)}$

5.2. se $|u_{nn}| \leq tol$ então terminar, a matriz é singular e $\lambda_n \leftarrow 0$.

5.3. calcular $w_n^{(k+1)} = \frac{aux_n}{u_{nn}}$

5.4. para $i = n-1, \ldots, 1$ calcular $soma = \sum_{l=i+1}^{n} u_{il} w_l^{(k+1)}$ e $w_i^{(k+1)} = \frac{aux_i - soma}{u_{ii}}$

6. para $i = 1, \ldots, n$ calcular $elem$ tal que $|w_{elem}^{(k+1)}| \geq |w_i^{(k+1)}|$

7. para $i = 1, \ldots, n$ calcular $v_i^{(k+1)} = \frac{w_i^{(k+1)}}{w_{elem}^{(k+1)}}$

8. fazer $\lambda^{(k+1)} = \frac{1}{w_{elem}^{(k+1)}}$ e para $i = 1, \ldots, n$ fazer $x_i^{(k+1)} = v_i^{(k+1)}$

9. se convergência=.FALSE. (algoritmo 5.1) então

9.1. se $k \geq n_{max}$ então recomeçar, ir para o passo 3

9.2. ir para o passo 4

10. terminar com $\lambda_n \leftarrow \lambda^{(k+1)}$ e $x_n \leftarrow (v_i^{(k+1)}, i = 1, \ldots, n)$.

A convergência da iteração inversa do método da potência é tanto mais rápida quanto menores forem as quantidades $\frac{\lambda_n}{\lambda_i}$, para $i = 1, \ldots, n-1$. Tal como foi feito em 5.3.1,

$$w^{(k+1)} = \sum_{i=1}^{n} \alpha_i (\lambda_i^{-1})^k x_i$$

e que pode ser escrita na forma

$$w^{(k+1)} = (\lambda_n^{-1})^k (\alpha_n x_n + \sum_{i=1}^{n-1} \alpha_i (\frac{\lambda_n}{\lambda_i})^k x_i), \tag{5.16}$$

donde se conclui que quanto menor for a contribuição do segundo termo do lado direito desta igualdade, mais o termo múltiplo de x_n domina e mais rápida é a convergência de $w^{(k+1)}$ em direcção a um múltiplo de x_n.

O que foi dito em 5.3.1 em relação aos valores complexos também se aplica na iteração inversa.

5.3.4 Implementação. Rotina MENVAL

A rotina MENVAL implementa o algoritmo 5.3. No critério de paragem (algoritmo 5.1) são usados os seguintes valores: $\varepsilon_1 = \varepsilon_2 = 10^{-6}$.

Exemplo 5.5 A matriz do exemplo 5.1 é singular. A implementação da iteração inversa (5.15) é baseada numa matriz U também singular, donde se conclui que o valor próprio de menor módulo é igual a zero.

Exemplo 5.6 O valor próprio de menor módulo da matriz do exemplo 5.2 é obtido ao fim de 20 iterações. Algumas iterações do processo são:

k	$\lambda^{(k+1)}$	$v^{(k+1)}$	k	$\lambda^{(k+1)}$	$v^{(k+1)}$
1	-0.181818	$(1,-0.363636)^T$	10	-0.999120	$(1,-0.499853)^T$
5	-0.971061	$(1,-0.495177)^T$	15	-0.999973	$(1,-0.499995)^T$

O último valor calculado é $\lambda_n = -0.999999$ com $x_n = (1.000000, -0.5)^T$. O processo foi lento. Repare-se que o quociente $\frac{\lambda_n}{\lambda_1} \approx 0.5$.

5.3.5 Valor próprio mais próximo de uma estimativa. Convergência

A iteração inversa e deslocada do método da potência serve para calcular o valor próprio de A que está mais próximo de uma estimativa est, fornecida por nós. Se $\lambda_1, \ldots, \lambda_i, \ldots, \lambda_n$, são os valores próprios de A então os valores próprios da matriz $A - est\,I$ são $\lambda_1 - est, \ldots, \lambda_i - est, \ldots, \lambda_n - est$. Quanto menor for a diferença $\lambda_i - est$, mais próximo λ_i está de est. Para calcular o valor próprio de A que está mais próximo de est, usa-se a matriz $A - est\,I$ e procura-se o seu valor próprio de menor valor absoluto, usando para isso a iteração inversa, como se fez em 5.3.3. Assim, a equação iterativa é

$$w^{(k+1)} = (A - est\,I)^{-1} v^{(k)}, \text{ para } k = 1, 2, \ldots \tag{5.17}$$

com $v^{(1)}$ vector arbitrário e

$$v^{(k)} = \frac{w^{(k)}}{|w^{(k)}|_\infty}$$

tendo $|w^{(k)}|_\infty$ sido já definido anteriormente. Esta equação pode ser escrita na forma de um sistema de equações lineares, em que a matriz dos coeficientes $(A - est\,I)$ é constante ao longo das iterações:

$$(A - est\,I) w^{(k+1)} = v^{(k)}, \text{ para } k = 1, 2, \ldots \tag{5.18}$$

Para a sua resolução este sistema é equivalente a

$$U w^{(k+1)} = J v^{(k)}, \text{ para } k = 1, 2, \ldots \tag{5.19}$$

em que as matrizes U e J são agora os factores de $A - est\,I$, de tal modo que

$$J(A - est\,I) = U.$$

Quanto mais próximo est estiver do valor exacto do valor próprio, mais perto da singularidade está a matriz $A - est\,I$ e portanto a sua matriz U. Podem surgir problemas na implementação prática de (5.19), pois é provável que u_{nn} esteja muito próximo de zero. Ultrapassa-se esta dificuldade, colocando no lugar de u_{nn}, uma quantidade pequena e positiva, digamos ε, e resolvendo o sistema resultante. O passo seguinte consiste em multiplicar todos os elementos do vector solução por ε - qualquer múltiplo de um vector próprio é ainda um vector próprio - Por último, calcula-se o limite de cada elemento do

vector quando ε tende para zero. O algoritmo 5.4, que corresponde a esta iteração inversa e deslocada, resolve esta questão da quase singularidade de uma maneira um pouco diferente, uma vez que o objectivo é a sua implementação directa (em VALVEC) no computador: se u_{nn} for, em valor absloluto, menor do que a tolerância ε_2, substitui-se o seu valor por ε_2 e continua-se a implementação. Um valor aceitável para ε_2 parece ser 10^{-4}.

Tal como nos dois casos anteriores, tem-se

$$|w^{(k)}|_\infty \to \frac{1}{\lambda_i - est}, \quad w^{(k)} \to \text{múltiplo de } x_i$$

$$\text{e } v^{(k)} \to x_i, \text{ quando } k \to \infty,$$

sendo λ_i o valor próprio de A que está mais próximo de est e x_i o vector a ele associado.

Podem surgir algumas dificuldades na implementação da equação iterativa (5.19), uma vez que $w^{(k+1)}$ pode apresentar-se deficitário da contribuição de x_i. Resolve-se facilmente esta falha, atribuindo ao vector $Jv^{(1)}$, da primeira iteração, o vector $(1, 1, \ldots, 1)^T$.

Algoritmo 5.4 :

1. ler n, n_{max}, ε_1, ε_2, e para $i = 1, \ldots, n$ ler $(a_{il}, l = 1, \ldots, n)$
2. ler est e fazer $k = 0$
3. para $i = 1, \ldots, n$ e $l = 1, \ldots, n$
   - 3.1. se $l = i$ então fazer $b_{ii} = a_{ii} - est$
   - 3.2. senão fazer $b_{il} = a_{il}$
4. calcular os factores J e U da matriz $(A - estI) = (b_{il})$, $i, l = 1, \ldots, n$, tais que $J(A - estI) = U$ (algoritmo 3.5, com selec="J e U")
5. se $|u_{nn}| < \varepsilon_2$ então fazer $u_{nn} = \varepsilon_2$
6. fazer $k = k + 1$
7. resolver o sistema $Uw^{(k+1)} = Jv^{(k)}$ em ordem a $w^{(k+1)}$:
   - 7.1. se $k = 1$ então para $i = 1, \ldots, n$ fazer $aux_i = 1$
   - 7.2. senão para $i = 1, \ldots, n$ calcular $aux_i = \sum_{l=1}^{n} j_{il} v_l^{(k)}$
   - 7.3. calcular $w_n^{(k+1)} = \frac{aux_n}{u_{nn}}$
   - 7.4. para $i = n-1, \ldots, 1$ calcular $soma = \sum_{l=i+1}^{n} u_{il} w_l^{(k+1)}$ e $w_i^{(k+1)} = \frac{aux_i - soma}{u_{ii}}$
8. para $i = 1, \ldots, n$ calcular $elem$ tal que $|w_{elem}^{(k+1)}| \geq |w_i^{(k+1)}|$
9. para $i = 1, \ldots, n$ calcular $v_i^{(k+1)} = \frac{w_i^{(k+1)}}{w_{elem}^{(k+1)}}$
10. estimar o valor próprio correspondente (pelo quociente de Rayleigh):
    - 10.1. para $i = 1, \ldots, n$ calcular $aux_i = \sum_{l=1}^{n} v_l^{(k+1)} a_{li}$

10.2. calcular $den = \sum_{i=1}^{n} (v_i^{(k+1)})^2$ e $num = \sum_{i=1}^{n} aux_i v_i^{(k+1)}$

10.3. calcular $\lambda^{(k+1)} = \frac{num}{den}$

11. para $i = 1, \ldots, n$ calcular $r_i = \sum_{l=1}^{n} a_{il} v_l^{(k+1)} - \lambda^{(k+1)} v_i^{(k+1)}$

12. calcular $norma = \sqrt{\sum_{i=1}^{n} r_i^2}$

13. se $norma > \varepsilon_1$ então

13.1. se $k \geq n_{max}$ então recomeçar, ir para o passo 2

13.2. ir para o passo 6

14. terminar com $\lambda_{est} \leftarrow \lambda^{(k+1)}$ e $x_{est} \leftarrow (v_i^{(k+1)}, i = 1, \ldots, n)$.

A rapidez de convergência da equação iterativa (5.17) depende das quantidades $\frac{\lambda_i - est}{\lambda_j - est}$, para $j = 1, \ldots, n$, com $j \neq i$. De (5.17) pode-se exprimir

$$\begin{aligned} w^{(k+1)} &= ((A - est\, I)^{-1})^k v^{(1)} \\ &= \sum_{j=1}^{n} \alpha_j (\lambda_j - est)^{-k} x_j \\ &= (\lambda_i - est)^{-k} (\alpha_i x_i + \sum_{j=1, j\neq i}^{n} \alpha_j (\frac{\lambda_i - est}{\lambda_j - est})^k x_j) \end{aligned}$$

donde se conclui que a convergência é tanto mais rápida quanto menor for a contribuição do segundo termo desta igualdade, ou seja, quanto maior for o denominador, $\lambda_j - est$, do quociente da expressão. Isto exige que os outros valores próprios, que não o λ_i que se procura, estejam o mais afastados possível da quantidade est.

Quando est é já uma boa aproximação ao valor próprio procurado, λ_i, e se conhece uma aproximação ao vector próprio associado, x_i, pode-se melhorar ainda mais a aproximação ao valor próprio usando o *quociente de Rayleigh* definido por

$$\lambda_{iR} = \frac{x_i^T A x_i}{x_i^T x_i}.$$

Para verificar a precisão com que os resultados foram obtidos, pode-se usar o vector *resíduo* r, definido por

$$r = Ax_i - \lambda_i x_i$$

e exigir que a sua norma, $\|r\|_2$, atinja um valor próximo de zero, por exemplo, seja menor ou igual do que a quantidade ε_1, pequena e positiva.

5.3.6 Implementação. Rotina VALVEC

A implementação do algoritmo 5.4 origina a rotina VALVEC. O processo iterativo é parado com o valor de ε_1 igual a 10^{-6}. Para ε_2 foi escolhido o valor 10^{-4}.

Exemplo 5.7 A matriz do exemplo 5.1 é singular e o valor próprio de menor módulo é nulo. Para calcular o vector próprio associado, usa-se a rotina VALVEC, com $est = 0.005$. Ao fim de 3 iterações a rotina fornece $\lambda = 0.000000008$ e o vector associado $(0.000000023, 0.000000008, 1.000000000)^T$.

Quando se pretende convergir para o valor próprio mais perto de 2, faz-se $est = 2.00985$ e ao fim de 9 iterações obtém-se $\lambda = 2.00109565$ e $x = (0.998904954, 1.000000000, 0.499726237)^T$, com $\|r\|_2 = \|Av^{(k+1)} - \lambda v^{(k+1)}\|_2 = 0.000001$. Na terceira iteração já se tem $\lambda = 2.003268037$, $x = (0.996737292, 1.000000000, 0.499184322)^T$ e $\|r\|_2 = 0.000008$, o que caracteriza um processo muito lento. Não esquecer que a matriz A tem dois valores próprios iguais a 2.

Exemplo 5.8 A matriz

$$A = \begin{pmatrix} 5 & -8 \\ -4 & 1 \end{pmatrix}$$

tem dois valores próprios distintos. A rotina VALVEC, para $est = -2$ calcula, ao fim de 7 iterações, $\lambda = -3$ e $x = (1, 1)^T$. Para $est = 8$, calcula $\lambda = 9$ e $x = (1, -0.5)^T$ ao fim de 8 iterações.

Exemplo 5.9 A matriz

$$A = \begin{pmatrix} 5 & -1 & -2 \\ -1 & 3 & -2 \\ -2 & -2 & 5 \end{pmatrix}$$

tem um valor próprio próximo de 5. Usando $est = 4.5$, ao fim de 9 iterações obtém-se $\lambda = 5$ e $x = (1, -1, 0.5)^T$.

5.4 Valores próprios de uma matriz simétrica

5.4.1 Introdução teórica

Dada uma matriz A, quadrada de ordem n, um dos objectivos das *transformações semelhantes*

$$Q^{-1}AQ = B$$

é escolher uma matriz Q de tal forma que a matriz B se torne a mais simples possível. O cálculo dos valores e vectores próprios de B é assim facilitado e rápido. A escolha de $Q = X$, em que X é a matriz cujas colunas são os vectores próprios de A, transformando A em D, diagonal, com os valores próprios sobre a diagonal principal, é possível, desde que A seja simétrica[3]. No entanto, é um processo complicado e moroso. Quando a matriz A não é simétrica, a diagonalização, $X^{-1}AX = D$, nem sempre é possível. Já se viu em 5.1.3, a relação existente entre os valores próprios e os vectores associados, de A e os de B.

São apresentadas, em 5.4.2 e 5.4.4, duas técnicas para o cálculo do conjunto inteiro dos valores próprios, baseadas em transformações semelhantes, que transformam uma matriz

[3]Teorema: Uma matriz real e simétrica admite apenas valores próprios reais.

A simétrica numa tridiagonal também simétrica. A transformação de Givens que faz uso de matrizes de rotação ortogonais e a transformação de Householder baseada em matrizes ortogonais e simétricas.

5.4.2 Transformação de Givens. Forma tridiagonal simétrica

Seja A uma matriz real e simétrica de orden n. Os seus valores próprios são reais e neste caso é possível construir matrizes ortogonais que transformam A numa matriz diagonal contendo os valores próprios sobre a diagonal. Este processo exige uma sequência infinita de transformações semelhantes ortogonais e é baseado em matrizes de rotação de Jacobi com o seguinte aspecto:

$$Q = \begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & cos(\theta) & \dots & sen(\theta) & & \\ & & & 1 & & & \\ & & & & \ddots & & \\ & & & & & 1 & \\ & & -sen(\theta) & \dots & cos(\theta) & & \\ & & & & & & \ddots \\ & & 0 & & & & 1 \end{pmatrix},$$

em que $|\theta| \leq \pi$. Quando se aplica esta matriz a um vector no plano, este sofre uma rotação de um ângulo igual a θ.

O *método de Givens*[4] baseado nas matrizes de rotação no plano, semelhantes às de Jacobi, transforma uma matriz real e simétrica à forma tridiagonal simétrica. Se os elementos $cos(\theta)$ forem colocados nas posições (i,i) e (j,j), $sen(\theta)$ na posição (i,j) e $-sen(\theta)$ na posição (j,i), o produto de A por Q, à direita, só altera as colunas i e j de A. Por sua vez, o produto de $Q^{-1} = Q^T$, à esquerda, por A, forma combinações das linhas i e j. Assim, se $A = (a_{rs})$ e $Q^T AQ = (\bar{a}_{rs})$, $(r, s = 1, \dots, n)$, então

[4] J. W. Givens Jr. viveu entre 1910 e 1993. O seu nome tornou-se conhecido pelas matrizes de rotação no plano que surgem no cálculo dos valores próprios. Esta técnica foi desenvolvida durante os anos 50. A sua carreira universitária começou na Universidade de Kentuchy, continuou na de Virginia e acabou na Universidade de Princeton onde recebeu o seu Ph.D. em 1936. Durante o período de 1937 a 1964 ensinou em várias Universidades dos E.U.A., altura em que foi trabalhar para o Laboratório Nacional de Argonne como Director da Divisão de Matemática Aplicada. Os seus domínios de interesse foram a Álgebra Linear Numérica e as Equações com Derivadas Parciais. Mesmo antes do termo 'software' matemático ter sido inventado, Givens sugeriu a implementação de algoritmos de interesse genérico colocando-os à disposição dos cientistas e engenheiros. Reconheceu a importância dos algoritmos da Álgebra Linear Numérica na linguagem ALGOL que entretanto apareceram na revista Numerische Mathematik e juntamente com J.H. Wilkinson iniciou um projecto que implementava esses algoritmos em Fortran. A sua distribuição foi concretizada através de uma das primeiras bibliotecas de 'software' matemático, a AMDLIB. Em 1969, Givens exerceu, durante dois anos, as funções de Presidente da SIAM (Sociedade Internacional de Matemática Aplicada). Em 1975 reformou-se de Argonne e voltou para a Universidade do Northwestern como professor de matemática (SIAM NEWS, Julho de 1993).

$$\bar{a}_{ri} = cos(\theta)a_{ri} - sen(\theta)a_{rj}, \quad \bar{a}_{rj} = sen(\theta)a_{ri} + cos(\theta)a_{rj}, \tag{5.20}$$

$$\bar{a}_{is} = cos(\theta)a_{is} - sen(\theta)a_{js}, \quad \bar{a}_{js} = sen(\theta)a_{is} + cos(\theta)a_{js}, \tag{5.21}$$

$$\bar{a}_{rs} = a_{rs} \quad \text{para} \quad r, s = 1, \ldots, n \text{ e } r, s \neq i, j\ ,$$

$$\bar{a}_{ii} = cos^2(\theta)a_{ii} - 2cos(\theta)sen(\theta)a_{ij} + sen^2(\theta)a_{jj},$$

$$\bar{a}_{ij} = cos(\theta)sen(\theta)(a_{ii} - a_{jj}) = (cos^2(\theta) - sen^2(\theta))a_{ij} = \bar{a}_{ji},$$

$$\bar{a}_{jj} = sen^2(\theta)a_{ii} + 2cos(\theta)sen(\theta)a_{ij} + cos^2(\theta)a_{jj}.$$

O ângulo θ deve ser escolhido de tal forma que a transformação que opera nas linhas e colunas i e j produz zeros nas posições $(i-1, j)$ e $(j, i-1)$. Assim, por exemplo,

- para a primeira etapa, pretende-se reduzir a zero os elementos a_{31} e a_{13}, o que faz com que $i = 2$ e $j = 3$. De (5.20) e (5.21) (com $r = 1$ e $s = 1$) tira-se que

$$cos(\theta) = \frac{a_{12}}{\sqrt{a_{12}^2 + a_{13}^2}} \quad \text{e} \quad sen(\theta) = -\frac{a_{13}}{\sqrt{a_{12}^2 + a_{13}^2}};$$

  o produto $Q_1^T A$ opera sobre linhas e só altera as linhas 2 e 3 da matriz A (anulando o a_{31}); o produto AQ_1, que opera sobre colunas, altera apenas os elementos das colunas 2 e 3 de A (anulando o a_{13}). Considerando $A_1 = A$,

$$A_2 = Q_1^T A_1 Q_1$$

$$\begin{pmatrix} * & * & 0 & \ldots & * \\ * & * & * & \ldots & * \\ 0 & * & * & \ldots & * \\ & & & \ldots & \\ * & * & * & \ldots & * \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & \ldots & 0 \\ 0 & cos(\theta) & -sen(\theta) & \ldots & 0 \\ 0 & sen(\theta) & cos(\theta) & \ldots & 0 \\ & & & \ldots & \\ 0 & 0 & 0 & \ldots & 1 \end{pmatrix} \begin{pmatrix} * & * & * & \ldots & * \\ * & * & * & \ldots & * \\ * & * & * & \ldots & * \\ & & & \ldots & \\ * & * & * & \ldots & * \end{pmatrix} Q_1$$

  Utiliza-se o $*$ para indicar que o elemento da matriz, naquela posição, é diferente de zero.

- Na segunda etapa tenta-se anular os elementos das posições $(4, 1)$ e $(1, 4)$. Faz-se $i = 2$ e $j = 4$, bem como $r = 1$ e $s = 1$. O ângulo será agora determinado por forma a que

$$cos(\theta) = \frac{a_{12}}{\sqrt{a_{12}^2 + a_{14}^2}} \quad \text{e} \quad sen(\theta) = -\frac{a_{14}}{\sqrt{a_{12}^2 + a_{14}^2}};$$

  a operação $Q_2^T A_2$ altera apenas as linhas 2 e 4 (anulando o a_{41}), a operação $A_2 Q_2$ altera as colunas 2 e 4 (anulando o a_{14}) e

$$A_3 = Q_2^T A_2 Q_2$$

$$\begin{pmatrix} * & * & 0 & 0 & \dots & * \\ * & * & * & * & \dots & * \\ 0 & * & * & * & \dots & * \\ 0 & * & * & * & \dots & * \\ & & & & \dots & \\ * & * & * & * & \dots & * \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & \dots & 0 \\ 0 & cos(\theta) & 0 & -sen(\theta) & \dots & 0 \\ 0 & 0 & 1 & 0 & \dots & 0 \\ 0 & sen(\theta) & 0 & cos(\theta) & \dots & 0 \\ & & & & \dots & \\ 0 & 0 & 0 & 0 & \dots & 1 \end{pmatrix} \begin{pmatrix} * & * & 0 & * & \dots & * \\ * & * & * & * & \dots & * \\ 0 & * & * & * & \dots & * \\ & * & * & * & \dots & * \\ & & & & \dots & \\ * & * & * & * & \dots & * \end{pmatrix} Q_2$$

- Após as primeiras $n-2$ etapas, tem-se já a matriz transformada com zeros nas posições $(3,1),(4,1),\dots,(n,1)$ e $(1,3),(1,4),\dots,(1,n)$. Apareceram zeros em posições da $1^{a\cdot}$ coluna e $1^{a\cdot}$ linha.

- As próximas $n-3$ etapas vão criar zeros nas posições: $(4,2)$ e $(2,4)$ fazendo $i=3$ e $j=4$, $(5,2)$ e $(2,5)$ com $i=3$ e $j=5$, e assim sucessivamente, até às posições $(n,2)$ e $(2,n)$ com $i=3$ e $j=n$. Aparecem zeros em posições da $2^{a\cdot}$ coluna e $2^{a\cdot}$ linha e a matriz tem já o seguinte aspecto

$$A_{s+1} = Q_s^T A_s Q_s\,, \quad s = (n-2)+(n-3)$$

$$\begin{pmatrix} * & * & 0 & \dots & 0 \\ * & * & * & \dots & 0 \\ 0 & * & * & \dots & * \\ 0 & 0 & * & \dots & * \\ & & & \dots & \\ 0 & 0 & * & \dots & * \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & cos(\theta) & \dots & -sen(\theta) \\ 0 & 0 & 0 & \dots & 0 \\ & & & \dots & \\ 0 & 0 & sen(\theta) & \dots & cos(\theta) \end{pmatrix} \begin{pmatrix} * & * & 0 & \dots & 0 \\ * & * & * & \dots & * \\ 0 & * & * & \dots & * \\ 0 & 0 & * & \dots & * \\ & & & \dots & \\ 0 & * & * & \dots & * \end{pmatrix} Q_s\,.$$

- As próximas $n-4$ etapas introduzirão zeros nas posições $(5,3),(6,3),\dots,(n,3)$ e nas suas simétricas;

- ...

- Finalmente na última etapa, os únicos elementos a anular são $a_{n\,n-2}$ e $a_{n-2\,n}$.

Nenhum elemento anulado se altera de uma etapa para a seguinte e a matriz transformada tem a forma de uma matriz tridiagonal simétrica, como a apresentada em (5.23). O número total de etapas é de $\frac{1}{2}(n-1)(n-2)$. Os passos da transformação de Givens são apresentados no algoritmo 5.5. Este, além de calcular a matriz transformada, que designamos por C, fornece a matriz $Q = Q_1 Q_2 \dots Q_s$, com $s = \frac{1}{2}(n-1)(n-2)$, da transformação.

Algoritmo 5.5 :

1. ler n, calcular tol e para $i = 1, \ldots, n$ ler $(a_{ij}, j = 1, \ldots, n)$
2. para $i = 1, \ldots, n$ e $j = i, \ldots, n$
   - 2.1. se $j = i$ então fazer $id_{ii} = 1$
   - 2.2. senão fazer $id_{ij} = id_{ji} = 0$
3. para $k = 1, \ldots, n-2$
   - 3.1. fazer $i = k + 1$
   - 3.2 para $j = i + 1, \ldots, n$
     - 3.2.1. calcular $den = \sqrt{a_{ki}^2 + a_{kj}^2}$
     - 3.2.2. se $den \leq tol$ então incrementar o índice j e regressar ao passo 3.2.1
     - 3.2.3. calcular $cos = \frac{a_{ki}}{den}$ e $sen = -\frac{a_{kj}}{den}$
     - 3.2.4. para $l = 1, \ldots, n$ e $m = 1, \ldots, n$
       - 3.2.4.1. se $m = l$ então fazer $q_{ll} = 1$
       - 3.2.4.2. senão fazer $q_{lm} = 0$
     - 3.2.5. fazer $q_{ii} = q_{jj} = cos$, $q_{ij} = -sen$ e $q_{ji} = sen$
     - 3.2.6. para $l = 1, \ldots, n$ e $m = 1, \ldots, n$
       - 3.2.6.1. se $l \leq k$ então fazer $aux_{lm} = a_{lm}$
       - 3.2.6.2. senão calcular $aux_{lm} = \sum_{r=1}^{n} q_{lr} a_{rm}$
     - 3.2.7. fazer $q_{ij} = -q_{ij}$ e $q_{ji} = -q_{ji}$
     - 3.2.8. para $l = 1, \ldots, n$ e $m = 1, \ldots, n$
       - 3.2.8.1. se $m \leq k$ então fazer $a_{lm} = aux_{lm}$
       - 3.2.8.2. senão calcular $a_{lm} = \sum_{r=1}^{n} aux_{lr} q_{rm}$
       - 3.2.8.3. se $m \leq k$ então fazer $tr_{lm} = id_{lm}$
       - 3.2.8.4. senão calcular $tr_{lm} = \sum_{r=1}^{n} id_{lr} q_{rm}$
     - 3.2.9. para $l = 1, \ldots, n$ e $m = 1, \ldots, n$ fazer $id_{lm} = tr_{lm}$
4. terminar com $C \leftarrow ((a_{ij}, j = i-1, i, i+1,$ desde que $j \geq 1$ e $j \leq n), i = 1, \ldots, n)$ e $Q \leftarrow ((tr_{ij}, j = 1, \ldots, n), i = 1, \ldots, n)$.

5.4.3 Implementação. Rotina GIVENS

A implementação do algoritmo 5.5 origina a rotina GIVENS, que foi usada na transformação da matriz do seguinte exemplo:

Exemplo 5.10 A transformação da matriz A,

$$\begin{pmatrix} 4 & 3 & 2 & 1 \\ 3 & 4 & 3 & 2 \\ 2 & 3 & 4 & 3 \\ 1 & 2 & 3 & 4 \end{pmatrix}$$

à forma tridiagonal simétrica é feita em três etapas. Na primeira etapa, a matriz de transformação é

$$Q_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.832050 & -0.554700 & 0 \\ 0 & 0.554700 & 0.832050 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Na segunda etapa tem-se já, para Q_1Q_2 e A_3, respectivamente

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.801784 & -0.554700 & -0.222375 \\ 0 & 0.534522 & 0.832050 & -0.148250 \\ 0 & 0.267261 & 0 & 0.963624 \end{pmatrix} \quad \text{e} \quad \begin{pmatrix} 4 & 3.741657 & 0 & 0 \\ 3.741657 & 8.285714 & 1.482499 & 2.139558 \\ 0 & 1.482499 & 1.230769 & 1.027928 \\ 0 & 2.139558 & 1.027928 & 2.483516 \end{pmatrix}$$

e finalmente, na última etapa fica-se com

$$Q = Q_1Q_2Q_3 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.801784 & -0.498707 & 0.329293 \\ 0 & 0.534522 & 0.352029 & -0.768350 \\ 0 & 0.267261 & 0.792065 & 0.548821 \end{pmatrix} \quad \text{e}$$

$$A_4 = C = \begin{pmatrix} 4 & 3.741657 & 0 & 0 \\ 3.741657 & 8.285714 & 2.602981 & 0 \\ 0 & 2.602981 & 3.039587 & 0.225401 \\ 0 & 0 & 0.225401 & 0.674699 \end{pmatrix}.$$

5.4.4 Transformação de Householder. Forma tridiagonal simétrica

As matrizes de transformação Householder são ortogonais e simétricas, $Q^{-1} = Q^T = Q$.

Se a matriz A for simétrica de ordem n, as transformações semelhantes de Householder reduzem-na à forma tridiagonal simétrica, ao fim de $n-2$ etapas,

$$A_1 = A$$

$$A_{k+1} = Q_k A_k Q_k\,, \quad k = 1, 2, \ldots, n-2, \tag{5.22}$$

sendo A_{n-1} uma matriz tridiagonal simétrica, C, como a representada em (5.23).

A matriz de transformação Q_1, da primeira etapa, é definida por

$$Q_1 = I - 2ww^T$$

sendo w um vector de n elementos,

$$w = (0, w_2, w_3, \ldots, w_n)^T,$$

que satisfazem $w_2^2 + w_3^2 + \cdots + w_n^2 = 1$. Como se pretende reduzir a zero os elementos de A_1 que estão nas posições $(1,3), (1,4), \ldots, (1,n)$ definem-se

$$w_2^2 = \frac{1}{2}(1 \pm \frac{a_{12}}{\sqrt{a_{12}^2 + a_{13}^2 + \cdots + a_{1n}^2}}),$$

$$w_3 = \pm \frac{a_{13}}{2w_2\sqrt{a_{12}^2 + a_{13}^2 + \cdots + a_{1n}^2}},$$

$$\cdots$$

$$w_n = \pm \frac{a_{1n}}{2w_2\sqrt{a_{1n}^2 + a_{13}^2 + \cdots + a_{1n}^2}}.$$

Como A_1 é simétrica, o produto de Q_1, à esquerda, por A_1 reduz também a zero os elementos das posições $(3,1),(4,1),\ldots,(n,1)$, enquanto que o produto de A_1, por Q_1, à direita, reduz a zero os elementos, já mencionados, da $1^{a.}$ linha.

A matriz resultante, após a primeira etapa terá o seguinte aspecto

$$A_2 = Q_1 A_1 Q_1$$

$$\begin{pmatrix} * & * & 0 & \ldots & 0 \\ * & * & * & \ldots & * \\ 0 & * & * & \ldots & * \\ & & \ldots & & \\ 0 & * & * & \ldots & * \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & \ldots & 0 \\ 0 & 1-2w_2^2 & -2w_2w_3 & \ldots & -2w_2w_n \\ 0 & -2w_3w_2 & 1-2w_3^2 & \ldots & -2w_3w_n \\ & & & \ldots & \\ 0 & -2w_nw_2 & -2w_nw_3 & \ldots & 1-2w_n^2 \end{pmatrix} \begin{pmatrix} * & * & * & \ldots & * \\ * & * & * & \ldots & * \\ * & * & * & \ldots & * \\ & & & \ldots & \\ * & * & * & \ldots & * \end{pmatrix} Q_1.$$

Na segunda etapa da transformação de Householder, anulam-se os elementos nas posições $(2,4),(2,5),\ldots,(2,n)$ e simultaneamente os simetricamente colocados: $(4,2),(5,2)$, $\ldots$, $(n,2)$. A matriz de Householder $Q_2 = I - 2ww^T$ é definida para o vector $w = (0,0,w_3,w_4,\ldots,w_n)$, com $w_3^2 + \cdots + w_n^2 = 1$ e

$$w_3^2 = \frac{1}{2}(1 \pm \frac{a_{23}}{\sqrt{a_{23}^2 + a_{24}^2 + \cdots + a_{2n}^2}}),$$

$$w_4 = \pm \frac{a_{24}}{2w_3\sqrt{a_{23}^2 + a_{24}^2 + \cdots + a_{2n}^2}}),$$

$$\cdots$$

$$w_n = \pm \frac{a_{2n}}{2w_3\sqrt{a_{23}^2 + a_{24}^2 + \cdots + a_{2n}^2}}).$$

A matriz A_3 apresenta o seguinte aspecto

$$A_3 = Q_2 A_2 Q_2$$

$$\begin{pmatrix} * & * & 0 & \dots & 0 \\ * & * & * & \dots & 0 \\ 0 & * & * & \dots & * \\ & & & \dots & \\ 0 & 0 & * & \dots & * \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1-2w_3^2 & \dots & -2w_3w_n \\ & & & \dots & \\ 0 & 0 & -2w_nw_3 & \dots & 1-2w_n^2 \end{pmatrix} \begin{pmatrix} * & * & 0 & \dots & 0 \\ * & * & * & \dots & * \\ 0 & * & * & \dots & * \\ & & & \dots & \\ 0 & * & * & \dots & * \end{pmatrix} Q_2.$$

O processo continua até se atingir a última etapa onde se devem anular os elementos $a_{n\,n-2}$ e $a_{n-2\,n}$ da matriz A_{n-2}. O vector w que compõe a matriz Q_{n-2} é definido por

$$w = (0, 0, \dots, 0, w_{n-1}, w_n)^T, \text{ com } w_{n-1}^2 + w_n^2 = 1$$

sendo

$$w_{n-1}^2 = \frac{1}{2}(1 \pm \frac{a_{n-2\,n-1}}{\sqrt{a_{n-2\,n-1}^2 + a_{n-2\,n}^2}})$$

e

$$w_n = \pm \frac{a_{n-2\,n}}{2w_{n-1}\sqrt{a_{n-2\,n-1}^2 + a_{n-2\,n}^2}}.$$

A matriz $A_{n-1} = Q_{n-2}A_{n-2}Q_{n-2}$ apresenta a forma tridiagonal simétrica. O algoritmo 5.6 apresenta todos os passos necessários para transformar uma matriz A simétrica numa tridiagonal, também simétrica, usando a técnica de transformação Householder. Fornece também a matriz da transformação $Q = Q_1Q_2 \dots Q_s$, com $s = n - 2$.

Algoritmo 5.6 :

1. ler n, calcular tol e para $i = 1, \dots, n$ ler $(a_{ij}, j = 1, \dots, n)$
2. para $i = 1, \dots, n$ e $j = i, \dots, n$
   - 2.1. se $j = i$ então fazer $id_{ii} = 1$
   - 2.2. senão fazer $id_{ij} = id_{ji} = 0$
3. para $k = 1, \dots, n-2$
   - 3.1. calcular $den = \sqrt{\sum_{j=k+1}^{n} a_{kj}^2}$
   - 3.2. se $den \leq tol$ então incrementar o índice k e regressar ao passo 3.1
   - 3.3. para $i = k+1, \dots, n$ fazer
     - 3.3.1. se $i = k+1$ então calcular $w_i = \sqrt{\frac{1}{2}(1 + sinal(a_{k\,k+1})\frac{a_{k\,k+1}}{den})}$
     - 3.3.2. senão calcular $w_i = sinal(a_{k\,k+1})\frac{a_{ki}}{2w_{k+1}den}$
   - 3.4. para $i = 1, \dots, n$ fazer
     - 3.4.1. se $i \leq k$ então fazer $q_{ii} = 1$ senão fazer $q_{ii} = 1 - 2w_i^2$
     - 3.4.2. para $j = i+1, \dots, n$
       - 3.4.2.1. se $i \leq k$ então fazer $q_{ij} = q_{ji} = 0$
       - 3.4.2.2. senão fazer $q_{ij} = q_{ji} = -2w_iw_j$

3.5. para $i = 1, \ldots, n$ e $j = 1, \ldots, n$ calcular $prod_{ij} = \sum_{l=1}^{n} q_{il} a_{lj}$
3.6. para $i = 1, \ldots, n$ e $j = 1, \ldots, n$ calcular $a_{ij} = \sum_{l=1}^{n} prod_{il} q_{lj}$
3.7. para $i = 1, \ldots, n$ e $j = 1, \ldots, n$ calcular $tr_{ij} = \sum_{l=1}^{n} id_{il} q_{lj}$
3.8. para $i = 1, \ldots, n$ e $j = 1, \ldots, n$ fazer $id_{ij} = tr_{ij}$

4. terminar com $C \leftarrow ((a_{ij}, j = i-1, i, i+1,$ desde que $j \geq 1$ e $j \leq n), i = 1, \ldots, n)$ e $Q \leftarrow ((tr_{ij}, j = 1, \ldots, n), i = 1, \ldots, n)$.

5.4.5 Implementação. Rotina HOUSEH

A rotina HOUSEH implementa o algoritmo 5.6. A matriz A é transformada em C pelas transformações de Householder.

Exemplo 5.11 A partir da matriz do exemplo 5.10, a rotina HOUSEH fornece, em duas etapas, as seguintes matrizes

$$Q_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -0.801784 & -0.534522 & -0.267261 \\ 0 & -0.534522 & 0.841427 & -0.079287 \\ 0 & -0.267261 & -0.079287 & 0.960357 \end{pmatrix},$$

$$A_2 = \begin{pmatrix} 4 & -3.741657 & 0 & 0 \\ -3.741657 & 8.285714 & -1.301430 & -2.254283 \\ 0 & -1.301430 & 1.070669 & 0.911284 \\ 0 & -2.254283 & 0.911284 & 2.643617 \end{pmatrix},$$

e finalmente

$$Q = Q_1 Q_2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -0.801784 & 0.498707 & 0.329293 \\ 0 & -0.534522 & -0.352029 & -0.768350 \\ 0 & -0.267261 & -0.792065 & 0.548821 \end{pmatrix}$$

$$\text{com} \quad A_3 = C = \begin{pmatrix} 4 & -3.741657 & 0 & 0 \\ -3.741657 & 8.285714 & 2.602981 & 0 \\ 0 & 2.602981 & 3.039587 & -0.225401 \\ 0 & 0 & -0.225401 & 0.674699 \end{pmatrix}.$$

5.4.6 Sequência de Stürm para a localização dos valores próprios

Considere-se, agora, o problema do cálculo dos valores próprios de uma matriz tridiagonal simétrica, com a forma

$$C = \begin{pmatrix} d_1 & b_2 & & & & & \\ b_2 & d_2 & b_3 & & & & \\ & \ddots & \ddots & \ddots & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & b_{n-1} & d_{n-1} & b_n \\ & & & & & b_n & d_n \end{pmatrix}. \tag{5.23}$$

Os determinantes das submatrizes principais de $C-\lambda I$, sendo λ um dos valores próprios de C, podem ser calculados, um de cada vez, usando o teorema de Laplace[5]. Fazendo o desenvolvimento ao longo dos dois elementos não nulos da última linha, gera-se a seguinte relação de recorrência

$$f_0(\lambda) = 1$$

$$f_1(\lambda) = d_1 - \lambda$$

$$f_k(\lambda) = (d_k - \lambda) f_{k-1}(\lambda) - b_k^2 f_{k-2}(\lambda)\,, \text{ para } k = 2, \ldots, n. \tag{5.24}$$

A sequência $f_0(\lambda), f_1(\lambda), \ldots, f_n(\lambda)$ forma uma *sequência de Stürm*. Podem-se utilizar as propriedades da sequência $\{f_k(\lambda)\}$ para localizar as raízes de $f_n(\lambda) = det(C - \lambda I) = 0$, ou seja, os valores próprios de C (relembrar o Teorema de Stürm em 2.4.3 do Capítulo 2). Como todos os valores próprios de uma matriz verificam

$$-\|C\|_\infty \leq \lambda_i \leq \|C\|_\infty$$

para $i = 1, \ldots, n$, dividindo ao meio este intervalo, pode-se calcular o número de raízes dentro de cada subintervalo. A bissecção, dos subintervalos resultantes, pode ser feita repetidas vezes até todas as raízes serem localizadas.

Localizada a raiz, a etapa seguinte consiste em melhorar a aproximação, usando o método de Newton do Capítulo 2 (veja-se em 2.8). Assim, dada uma aproximação inicial ao valor próprio, $\lambda^{(1)}$, que pode ser o ponto médio do último intervalo de localização, obtém-se uma sequência de aproximações ao valor próprio, usando a equação iterativa

$$\lambda^{(k+1)} = \lambda^{(k)} - \frac{f_n(\lambda^{(k)})}{f_n'(\lambda^{(k)})}\,, \text{ para } k = 1, 2, \ldots$$

em que $f_n(\lambda^{(k)})$ é o último elemento da sequência (5.24), quando $\lambda = \lambda^{(k)}$. O valor de $f_n'(\lambda^{(k)})$ obtém-se por um processo de recorrência idêntico. O último elemento da sequência

$$f_0'(\lambda) = 0$$

$$f_1'(\lambda) = -1$$

$$f_k'(\lambda) = (d_k - \lambda) f_{k-1}'(\lambda) - b_k^2 f_{k-2}'(\lambda) - f_{k-1}(\lambda)\,, \text{ para } k = 2, \ldots, n, \tag{5.25}$$

dá, quando $\lambda = \lambda^{(k)}$, o valor de $f_n'(\lambda^{(k)})$. O algoritmo 5.7 utiliza o Teorema de Stürm para localizar e calcular aproximações aos valores próprios de uma matriz tridiagonal. Na segunda parte do algoritmo (passo 5) essas aproximações são melhoradas pelo método de Newton.

[5]Teorema de Laplace: Um determinante é igual à soma algébrica dos produtos dos elementos de uma linha pelos respectivos complementos algébricos.

Algoritmo 5.7 :

1. ler n, ε, n_{max}, calcular tol, para $i = 1, \ldots, n$ ler d_i e para $i = 2, \ldots, n$ ler b_i
2. calcular $s_1 = |d_1| + |b_2|$, $s_n = |d_n| + |b_n|$ e para $i = 2, \ldots, n-1$ calcular $s_i = |b_i| + |d_i| + |b_{i+1}|$
3. calcular $norma = máx(s_1, s_2, \ldots, s_n)$
4. localizar os valores próprios de C:
   para $r = 1, \ldots, n$
   - 4.1. fazer $est = 0$
   - 4.2. fazer $m = \frac{norma}{2}$, $f_0 = 1$ e $sin_0 = 1$
   - 4.3. verificar as concordâncias de sinal da sequência de Stürm:
     - 4.3.1. fazer $ncs = 0$
     - 4.3.2. calcular $f_1 = d_1 - est$ e fazer $sin_1 = sinal(f_1)$
     - 4.3.3. se $sin_1 = sin_0$ então fazer $ncs = ncs + 1$
     - 4.3.4. para $k = 2, \ldots, n$
       - 4.3.4.1. calcular $f_k = (d_k - est)f_{k-1} - b_k^2 f_{k-2}$ e fazer $sin_k = sinal(f_k)$
       - 4.3.4.2. se $sin_k = sin_{k-1}$ então fazer $ncs = ncs + 1$
     - 4.3.5. se $ncs - r < 0$ então fazer $est = est - m$
     - 4.3.6. senão fazer $est = est + m$
     - 4.3.7. fazer $m = \frac{m}{2}$
     - 4.3.8. se $m \geq \frac{\varepsilon}{2}$ então recomeçar a partir do passo 4.3
   - 4.4. guardar o valor estimado: $\bar{\lambda}_r \leftarrow est$
5. melhorar a aproximação de cada valor próprio, pelo método de Newton:
   para $r = 1, \ldots, n$
   - 5.1. fazer $i = 0$ e $\lambda^{(1)} = \bar{\lambda}_r$
   - 5.2. fazer $i = i + 1$, $f_0 = 1$, $f_0' = 0$, $f_1 = d_1 - \lambda^{(i)}$ e $f_1' = -1$
   - 5.3. para $k = 2, \ldots, n$ calcular $f_k = (d_k - \lambda^{(i)})f_{k-1} - b_k^2 f_{k-2}$ e $f_k' = (d_k - \lambda^{(i)})f_{k-1}' - b_k^2 f_{k-2}' - f_{k-1}$
   - 5.4. se convergência=.FALSE. (algoritmo 2.4) então
     - 5.4.1. se $i \geq n_{max}$ então terminar, o processo não converge (poderá recomeçar dando outro valor a $\lambda^{(1)}$ (passo 5.1)
     - 5.4.2. se $|f_n'| \leq tol$ então terminar, o processo não está a convergir
     - 5.4.3. calcular $\lambda^{(i+1)} = \lambda^{(i)} - \frac{f_n}{f_n'}$ e ir para o passo 5.2
   - 5.5. guardar o valor calculado: $\lambda_r \leftarrow \lambda^{(i)}$
6. terminar com λ_r , $r = 1, \ldots, n$.

5.4.7 Implementação. Rotina TSTURM

A rotina TSTURM surge da implementação do algoritmo 5.7. Os dois parâmetros usados no critério de paragem do método de Newton (algoritmo 2.4) são $\varepsilon_1 = 10^{-8}$ e $\varepsilon_2 = 10^{-6}$. Ao parâmetro ε da primeira parte do algoritmo 5.7 (etapa 4.3.8) é dado o valor 10^{-2}.

Exemplo 5.12 Dada a matriz tridiagonal simétrica

$$C = \begin{pmatrix} 3 & -4.1231 & 0 \\ -4.1231 & 1.3529 & 3.4118 \\ 0 & 3.4118 & 5.6471 \end{pmatrix}$$

a rotina TSTURM fornece as seguintes aproximações aos valores próprios $\bar{\lambda}_1 = 8.501565, \bar{\lambda}_2 = 4.379058$ e $\bar{\lambda}_3 = -2.892832$, que depois de melhoradas pelo método de Newton ficam $\lambda_1 = 8.508031$, $\lambda_2 = 4.386119$ e $\lambda_3 = -2.894150$.

Exemplo 5.13 Para a matriz tridiagonal simétrica

$$C = \begin{pmatrix} 1 & 2 & 0 \\ 2 & 1 & 0 \\ 0 & 0 & 3 \end{pmatrix}$$

a rotina TSTURM calcula as seguintes aproximações aos valores próprios $\bar{\lambda}_1 = 2.994141, \bar{\lambda}_2 = 2.994141$ e $\bar{\lambda}_3 = -1.001953$, que depois de melhoradas pelo método de Newton ficam $\lambda_1 = 3, \lambda_2 = 3$ e $\lambda_3 = -1$. O facto da matriz apresentar zeros nas posições $(3,2)$ e $(2,3)$, que não são necessariamente nulas numa tridiagonal, explica dois valores próprios iguais.

5.4.8 Cálculo dos vectores associados aos valores próprios

Para calcular os vectores próprios, $x_1, x_2, \ldots, x_n$, associados respectivamente aos valores próprios, $\lambda_1, \lambda_2, \ldots, \lambda_n$, resolvem-se as equações,

$$(C - \lambda_i I)x_i = 0$$

para cada i, $i = 1, \ldots, n$. Estas equações homogéneas têm uma solução não trivial, $x_i \neq 0$, se e só se $det(C - \lambda_i I) = 0$. No desenvolvimento do determinante obtém-se o *polinómio característico*, que é um polinómio de grau n, em λ,

$$\lambda^n + a_{n-1}\lambda^{n-1} + \cdots + a_0$$

cujos zeros são os n valores próprios de C. Na prática, este procedimento não é aconselhável. O cálculo dos coeficientes a_i é bastante longo e nada económico, especialmente para $n > 3$. Além disso, torna-se instável e os valores próprios calculados vêm afectados de grandes erros, pois pequenas perturbações nos coeficientes de um polinómio podem gerar grandes perturbações nos seus zeros (relembrar o que foi dito no Capítulo 2, em 2.2.2).

Assim, o método mais aconselhável para calcular o vector próprio, x_i, da matriz C, associado ao valor próprio λ_i, consiste na implementação da iteração inversa e deslocada do método da potência, como já foi referido em 5.3.5,

$$(C - \lambda_i I)w^{(k+1)} = v^{(k)}, \text{ para } k = 1, 2, \ldots$$

em que λ_i é o valor calculado anteriormente.

O algoritmo correspondente é o algoritmo 5.4 e a rotina é a VALVEC. Este processo gera o vector próprio x_i da matriz C, com grande precisão, desde que o processo iterativo seja repetido até se obter um vector resíduo, $Cx_i - \lambda_i x_i$, muito pequeno. A rotina VALVEC deve ser usada tantas vezes quantos os valores próprios da matriz C para os quais se quer o vector próprio associado.

Conhecidos os vectores próprios de C, associados aos valores λ_i, resta, agora, construir os vectores próprios da matriz original A. Assim, a última etapa consiste em calcular os vectores próprios da matriz A, $x_i(A)$, que correspondem aos vectores próprios de C, $x_i(C)$, usando a matriz de transformação Q,

$$x_i(A) = Qx_i(C)$$

sendo Q a matriz definida pelo produto das diversas matrizes de transformação das etapas,

$$Q = Q_1 Q_2 \dots Q_s$$

sendo s o número de etapas. s é igual a $\frac{1}{2}(n-1)(n-2)$, se é usada a transformação de Givens (veja-se em 5.4.2.) e igual a $n-2$, se é usada a transformação de Householder (veja-se em 5.4.4.). Tanto a rotina GIVENS (algoritmo 5.5) como a HOUSEH (algoritmo 5.6) fornecem a matriz da transformação Q.

5.5 Valores próprios de uma matriz não simétrica

5.5.1 Introdução

Quando a matriz A não é simétrica, o uso das matrizes de transformação Givens ou Householder, reduz A à forma quase-triangular superior, conhecida por *forma Hessenberg*, uma vez que os elementos de A nas posições denominadas simétricas são diferentes. Estas implementações necessitam de um número elevado de operações face ao objectivo a atingir. Torna-se mais rápida a implementação de *transformações elementares* que são baseadas em matrizes elementares semelhantes às usadas no processo de eliminação de Gauss quando da resolução de sistemas de equações lineares (veja-se em 3.4.2 e 3.4.4 do Capítulo 3).

5.5.2 Transformações elementares. Forma Hessenberg

Designe-se a matriz de transformação da etapa k por M_k. São precisas $n-2$ etapas para se chegar à forma Hessenberg superior. Cada etapa, k, reduz a zero um grupo de elementos da coluna k da matriz A_k. Assim, na primeira etapa, são reduzidos a zero os elementos $a_{31}, a_{41}, \dots, a_{n1}$. Na segunda etapa, são reduzidos a zero os elementos das posições $(4,2), (5,2), \dots, (n,2)$, e assim por adiante, até se chegar à última etapa, onde se reduz a zero o elemento $a_{n\,n-2}$.

O processo de transformação consiste em:

$$A_1 = A$$

$$A_{k+1} = M_k^{-1} A_k M_k \,, \text{ para } k = 1, 2, \ldots, n-2$$

sendo

$$A_{n-1} = H = \begin{pmatrix} * & * & * & \ldots & * & * \\ * & * & * & \ldots & * & * \\ 0 & * & * & \ldots & * & * \\ 0 & 0 & * & \ldots & * & * \\ & & & \ldots & & \\ 0 & 0 & 0 & \ldots & * & * \end{pmatrix}$$

e as matrizes M_k não são ortogonais, mas sim matrizes elementares, isto é, operam sobre as linhas de A_k quando estão a multiplicar à esquerda e, operam sobre as colunas de A_k quando estão a multiplicar à direita. As operações efectuadas são as operações elementares já referidas em 3.4.1 (Capítulo 3). Assim, as matrizes M_k têm o mesmo aspecto das matrizes J_i da decomposição de matrizes (veja-se em 3.5.1, equações (3.18) e (3.19)). Assim, por exemplo, na primeira etapa, para anular os elementos $a_{31}, a_{41}, \ldots, a_{n1}$, toma-se como elemento 'pivot' o elemento da posição $(2,1)$. Os multiplicadores, tal como se fez no Capítulo 3, são definidos por

$$m_{32} = -\frac{a_{31}}{a_{21}}, \; m_{42} = -\frac{a_{41}}{a_{21}}, \; \ldots, \; m_{n2} = -\frac{a_{n1}}{a_{21}},$$

originando a seguinte transformação

$$A_2 = M_1^{-1} A_1 M_1$$

$$\begin{pmatrix} * & * & * & \ldots & * \\ * & * & * & \ldots & * \\ 0 & * & * & \ldots & * \\ 0 & * & * & \ldots & * \\ & & & \ldots & \\ 0 & * & * & \ldots & * \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & \ldots & 0 \\ 0 & 1 & 0 & \ldots & 0 \\ 0 & m_{32} & 1 & \ldots & 0 \\ 0 & m_{42} & 0 & \ldots & 0 \\ & & & \ldots & \\ 0 & m_{n2} & 0 & \ldots & 1 \end{pmatrix} \begin{pmatrix} * & * & * & \ldots & * \\ * & * & * & \ldots & * \\ * & * & * & \ldots & * \\ * & * & * & \ldots & * \\ & & & \ldots & \\ * & * & * & \ldots & * \end{pmatrix} M_1 \,.$$

Para conservar a estabilidade deste processo há necessidade de colocar na posição 'pivot', que é a posição $(k+1, k)$ para a etapa k, o elemento de maior módulo, dentre os elementos $a_{k+1\,k}, a_{k+2\,k}, \ldots, a_{n\,k}$. Tal com foi feito no processo de eliminação de Gauss com pivotagem parcial, a escolha do 'pivot' mais adequado, pode levar a trocas de linhas antes do cálculo dos multiplicadores. Com já se sabe, este processo é equivalente à multiplicação, de A_k, à esquerda, por uma certa matriz elementar. Como se está perante um processo de tranformação semelhante, a matriz A_k deve ser também multiplicada à direita pela respectiva matriz elementar (a que corresponde a troca das correspondentes colunas). Nesta primeira etapa, e com pivotagem parcial, a transformação resume-se a

$$A_2 = M_1^{-1} I_{2,j} A_1 I_{2,j} M_1 \,, \text{ para } j = 3, 4, \ldots \text{ ou } n.$$

A segunda etapa tem como objectivo anular os elementos $a_{42}, a_{52}, \ldots, a_{n2}$ da matriz A_2 e consiste na seguinte transformação:

$$A_3 = M_2^{-1} I_{3,j} A_2 I_{3,j} M_2, \text{ para } j = 4, 5, \ldots \text{ ou } n$$

$$\begin{pmatrix} * & * & * & \ldots & * \\ * & * & * & \ldots & * \\ 0 & * & * & \ldots & * \\ 0 & 0 & * & \ldots & * \\ & & & \ldots & \\ 0 & 0 & * & \ldots & * \end{pmatrix} = M_2^{-1} I_{3,j} \begin{pmatrix} * & * & * & \ldots & * \\ * & * & * & \ldots & * \\ * & * & * & \ldots & * \\ 0 & * & * & \ldots & * \\ & & & \ldots & \\ 0 & * & * & \ldots & * \end{pmatrix} I_{3,j} \begin{pmatrix} 1 & 0 & 0 & \ldots & 0 \\ 0 & 1 & 0 & \ldots & 0 \\ 0 & 0 & 1 & \ldots & 0 \\ 0 & 0 & -m_{43} & \ldots & 0 \\ & & & \ldots & \\ 0 & 0 & -m_{n3} & \ldots & 1 \end{pmatrix},$$

com os multiplicadores calculados do seguinte modo,

$$m_{43} = -\frac{a_{42}}{a_{32}}, \; m_{53} = -\frac{a_{52}}{a_{32}}, \; \ldots, \; m_{n2} = -\frac{a_{n2}}{a_{32}}$$

baseados no 'pivot', que é o elemento da (ou que se colocou, através de trocas das linhas 3 e j, na) posição $(3, 2)$.

Este processo é repetido até se chegar à última etapa. Nesta, já só é preciso calcular o multiplicador $m_{n\,n-1} = -\frac{a_{n\,n-2}}{a_{n-1\,n-2}}$ que define uma matriz elementar, M_{n-2}, triangular inferior com $1's$ sobre a diagonal em que o único elemento diferente de zero, fora da diagonal, é o $-m_{n\,n-1}$ na posição $(n, n-1)$ e

$$H = A_{n-1} = M_{n-2}^{-1} I_{n-1\,,n}\, A_{n-2}\, I_{n-1\,,n}\, M_{n-2}.$$

A matriz A_{n-1}, após as $n-2$ etapas, tem a forma Hessenberg,

$$A_{n-1} = H = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \ldots & a_{1n-1} & a_{1n} \\ a_{21} & a_{22} & a_{23} & \ldots & a_{2n-1} & a_{2n} \\ 0 & a_{32} & a_{33} & \ldots & a_{3n-1} & a_{3n} \\ & & & \ldots & & \\ 0 & 0 & 0 & \ldots & a_{nn-1} & a_{nn} \end{pmatrix}.$$

A transformação semelhante e elementar descrita, baseada na matriz M, e que transforma uma matriz não simétrica à forma quase triangular superior, é apresentada no algoritmo 5.8. Este, fornece também a matriz da transformação $M = M_1 M_2 \ldots M_{n-2}$.

Algoritmo 5.8 :

1. `ler` n, `calcular` tol `e para` $i = 1, \ldots, n$ `ler` $(a_{ij}, j = 1, \ldots, n)$
2. `para` $i = 1, \ldots, n$ `e` $j = i, \ldots, n$
   - 2.1. `se` $j = i$ `então fazer` $tr_{ii} = 1$
   - 2.2. `senão fazer` $tr_{ij} = tr_{ji} = 0$

3. para $j = 1, \ldots, n-2$ fazer

   3.1. para $k = j+1, \ldots, n$ calcular piv tal que $|a_{piv\,j}| \geq |a_{kj}|$

   3.2. se $piv \neq j+1$ então

      3.2.1. para $k = j, \ldots, n$ trocar $a_{piv\,k}$ com $a_{j+1\,k}$

      3.2.2. para $k = 1, \ldots, n$ trocar $a_{k\,piv}$ com $a_{k\,j+1}$

      3.2.3. para $k = 1, \ldots, n$ trocar $tr_{k\,piv}$ com $tr_{k\,j+1}$

   3.3. se $|a_{j+1\,j}| \leq tol$ então terminar, a matriz A é singular

   3.4. para $i = j+2, \ldots, n$

      3.4.1. calcular $m_{i\,j+1} = -\frac{a_{ij}}{a_{j+1\,j}}$

      3.4.2. para $k = j+1, \ldots, n$ calcular $a_{ik} = a_{ik} + m_{i\,j+1} a_{j+1\,k}$

   3.5. para $k = 1, \ldots, n$

      3.5.1. calcular $a_{k\,j+1} = a_{k\,j+1} - \sum_{i=j+2}^{n} a_{ki} m_{i\,j+1}$

      3.5.2. calcular $tr_{k\,j+1} = tr_{k\,j+1} - \sum_{i=j+2}^{n} tr_{ki} m_{i\,j+1}$

4. terminar com $H \leftarrow ((a_{ij}, j = i-1, \ldots, n, j \neq 0), i = 1, \ldots, n)$ e $M \leftarrow ((tr_{ij}, j = 1, \ldots, n),\ i = 1, \ldots, n)$.

5.5.3 Implementação. Rotina HESSEN

A implementação das transformações elementares que reduzem uma matriz não simétrica A à forma Hessenberg (algoritmo 5.8) origina a rotina HESSEN.

Exemplo 5.14 Para a matriz

$$A = \begin{pmatrix} 0.987 & 0.4 & -0.487 \\ -0.079 & 0.5 & 0.479 \\ 0.082 & 0.4 & 0.418 \end{pmatrix}$$

a rotina HESSEN fornece as matrizes H e M (apenas $n-2=1$ etapa):

$$H = \begin{pmatrix} 0.987 & 0.905494 & -0.487 \\ -0.079 & 0.00281 & 0.479 \\ 0 & -0.030957 & 0.91519 \end{pmatrix}, M = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1.037975 & 1 \end{pmatrix}.$$

Exemplo 5.15 Para a matriz

$$A = \begin{pmatrix} 32 & 19 & 1 \\ -2 & 23 & -4 \\ 4 & 17 & 26 \end{pmatrix}$$

tem-se a matriz transformada

$$H = \begin{pmatrix} 32 & 17 & 1 \\ -2 & 31 & -4 \\ 0 & 27 & 18 \end{pmatrix}, \text{ sendo } M = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -2 & 1 \end{pmatrix}$$

ao fim de uma etapa.

5.5.4 Valores próprios de uma matriz não simétrica. Método Q-R

O método que aqui se descreve serve para calcular os valores próprios de uma matriz não simétrica, e em particular de uma matriz com a forma quase triangular superior. É um método iterativo que transforma uma matriz genérica A (ou H), noutra, com uma forma mais simples, ao fim de um número infinito de etapas. Esta forma mais simples é a triangular superior, e sobre a diagonal principal surgem os valores próprios de A (ou de H), ordenados, do maior ao menor, em valor absoluto.

O método, conhecido por *método Q-R* consiste no seguinte:

Seja $A_1 = A$ a matriz original. Usando matrizes ortogonais, do tipo Givens, já introduzidas neste Capítulo, em 5.4.2., ou do tipo Householder, também já introduzidas em 3.5.5, Capítulo 3, calcula-se a factorização $Q_1 R_1$ de A_1 (R_1 é uma matriz triangular superior e Q_1 é ortogonal). Em seguida, multiplicam-se estas matrizes pela ordem inversa, isto é, $R_1\, Q_1$, para dar uma nova matriz A_2. A partir desta A_2, calcula-se a sua factorização $Q_2\, R_2$, multiplicando-as em seguida pela ordem inversa para dar A_3. Este processo repete-se iterativamente até se obter uma matriz A_k triangular superior. A equação iterativa na sua forma geral é

$$A_k = Q_k\, R_k\,, \;\; \text{com} \;\; A_{k+1} = R_k Q_k\,, \;\; k = 1, 2, \ldots$$

Como $Q_k^{-1}\, A_k = R_k$ então $A_{k+1} = Q_k^{-1}\, A_k\, Q_k$ sendo cada etapa uma transformação semelhante. Assim, os valores próprios de A_k são iguais aos da matriz original A.

Pode-se mostrar, sob certas condições, não muito restritivas, que no caso dos valores próprios serem todos reais e distintos, tem-se

$$Q_k \to I\,, \;\; A_k \to R_k \to \begin{pmatrix} \lambda_1 & \times & \times & \ldots & \times & \times \\ 0 & \lambda_2 & \times & \ldots & \times & \times \\ 0 & 0 & \lambda_3 & \ldots & \times & \times \\ & & & \ldots & & \\ 0 & 0 & 0 & \ldots & \lambda_{n-1} & \times \\ 0 & 0 & 0 & \ldots & 0 & \lambda_n \end{pmatrix},$$

em que $|\lambda_1| > |\lambda_2| > \ldots > |\lambda_n|$. O sinal $\times$ significa que naquelas posições o valor é diferente de zero.

Quando a matriz A não é simétrica é de esperar valores próprios complexos e, nesta situação, a matriz A_k tende, quando $k \to \infty$, para uma matriz não exactamente triangular superior, mas sim uma matriz do tipo

$$A_k \to \begin{pmatrix} \times & \times & \times & \ldots & \times & \times \\ 0 & * & * & \ldots & \times & \times \\ 0 & * & * & \ldots & \times & \times \\ & & & \ldots & & \\ 0 & 0 & 0 & \ldots & \times & \times \\ 0 & 0 & 0 & \ldots & 0 & \times \end{pmatrix},$$

em que cada matriz de 2×2, que aparece sobre a diagonal, como a assinalada com os símbolos $*$, tem um par de valores complexos conjugados como valores próprios. A determinação deste par faz-se através da resolução de

$$det \begin{pmatrix} a_{ii} - \lambda & a_{i\,i+1} \\ a_{i+1\,i} & a_{i+1\,i+1} - \lambda \end{pmatrix} = 0\,, \tag{5.26}$$

sendo $a_{ii}, a_{i\,i+1}, a_{i+1\,i}$ e $a_{i+1\,i+1}$ os elementos de A_k que se encontram nas posições assinaladas com $*$.

Como este método iterativo Q-R pode envolver bastantes cálculos, aconselha-se em primeiro lugar, a redução de A à forma Hessenberg, H, e só então, a partir desta, a implementação do método Q-R. Em todas as etapas as matrizes conservam a forma Hessenberg, de tal forma que a determinação da matriz Q do tipo Householder vem muito simplificada.

O algoritmo 5.9 representa na primeira parte o método Q-R. Pressupõe que a matriz dada está na forma Hessenberg (dada pelo algoritmo 5.8) e termina o processo iterativo quando Q_k está próxima da identidade. A tolerância ϵ do algoritmo é uma quantidade positiva e próxima de zero que serve para verificar a proximidade de Q_k em relação à I. Um valor da ordem dos 10^{-8} é adequado.

A segunda parte do algoritmo (a partir do passo 9) faz uma pesquisa na matriz R_k, nomeadamente nas posições da diagonal principal e na diagonal abaixo da diagonal principal, para verificar quais os valores próprios reais e quais os complexos. Um valor, que esteja sobre a diagonal abaixo da diagonal principal, por exemplo $a_{i+1\,i}$, que verifique

$$|a_{i+1\,i}| < tol\,(|a_{ii}| + |a_{i+1\,i+1}|) \tag{5.27}$$

pode ser considerado insignificante e quase nulo. A quantidade *tol* foi já definida (algoritmo 2.5, Capítulo 2) como a precisão do computador. Nesta situação, existe um valor próprio real igual a $a_{i\,i}$. Se $a_{i+1\,i}$ tem um valor 'diferente de zero', determina-se o par de valores complexos conjugados usando a equação (5.26).

Algoritmo 5.9 :

1. `ler` n, ϵ, `calcular` *tol*, `fazer` $s = 0$ `e para` $i = 1, \ldots, n$ `ler` $(a_{ij}, j = i-1, \ldots, n$ `e` $j \neq 0)$
2. `para` $i = 1, \ldots, n$ `e` $j = i, \ldots, n$
   - 2.1. `se` $j = i$ `então fazer` $aux_{ii} = 1$
   - 2.2. `senão fazer` $aux_{ij} = aux_{ji} = 0$
3. `para` $k = 1, \ldots, n-1$
   - 3.1. `calcular` $den = \sqrt{a_{kk}^2 + a_{k+1\,k}^2}$
   - 3.2. `se` $den \leq tol$ `então incrementar o índice k e regressar ao passo 3.1`
   - 3.3. `calcular` $w_k = \sqrt{\frac{1}{2}(1 + sinal(a_{kk})\frac{a_{kk}}{den})}$ `e` $w_{k+1} = sinal(a_{kk})\frac{a_{k+1\,k}}{2w_k den}$

3.4. para $i = 1, \ldots, n$

3.4.1. se ($i = k$ ou $i = k+1$) então fazer $q_{ii} = 1 - 2w_i^2$ senão fazer $q_{ii} = 1$

3.4.2. para $j = i+1, \ldots, n$

3.4.2.1. se ($i = k$ e $j = k+1$) então fazer $q_{ij} = q_{ji} = -2w_i w_j$

3.4.2.2. senão fazer $q_{ij} = q_{ji} = 0$

3.5. para $i = 1, \ldots, n$ e $j = 1, \ldots, n$ calcular $prod_{ij} = \sum_{l=1}^{n} q_{il} a_{lj}$

3.6. para $i = 1, \ldots, n$ e $j = 1, \ldots, n$ calcular $ort_{ij} = \sum_{l=1}^{n} aux_{il} q_{lj}$

3.7. para $i = 1, \ldots, n$ e $j = 1, \ldots, n$ fazer $aux_{ij} = ort_{ij}$ e $a_{ij} = prod_{ij}$

4. para $i = 1, \ldots, n$ e $j = 1, \ldots, n$ calcular $a_{ij} = \sum_{l=1}^{n} prod_{il} ort_{lj}$

5. fazer $ident = .TRUE.$

6. para $i = 1, \ldots, n-2$ e $j = i+2, \ldots, n$
se $|ort_{ij}| > \epsilon$ ou $|ort_{ji}| > \epsilon$ então fazer $ident = .FALSE.$ e ir para o passo 7

7. se $ident = .FALSE.$ então regressar ao passo 2

8. fazer $i = 1$

9. enquanto $i \leq n-1$

9.1. se $|a_{i+1\,i}| < tol\,(|a_{ii}| + |a_{i+1\,i+1}|)$ então fazer $\lambda_i = a_{ii}$, $i = i+1$

9.2. senão calcular $y = a_{i+1\,i} a_{i\,i+1}$, $z = (a_{ii} - a_{i+1\,i+1})^2 + 4y$ e $y = \sqrt{|z|}$

9.2.1. se $z > 0$ então calcular $\lambda_i = \frac{a_{ii} + a_{i+1\,i+1} + y}{2}$ e $\lambda_{i+1} = \frac{a_{ii} + a_{i+1\,i+1} - y}{2}$ e fazer $i = i+2$

9.2.2. senão calcular $R = \frac{a_{ii} + a_{i+1\,i+1}}{2}$ e $IM = \frac{y}{2}$ e fazer $\lambda_i = R + IM\imath$, $\lambda_{i+1} = R - IM\imath$ e $i = i+2$

9.3. se $i = n$ então fazer $\lambda_i = a_{ii}$

10. terminar com $(\lambda_i, i = 1, \ldots, n)$.

5.5.5 Implementação. Rotina TRANQR

A rotina TRANQR implementa o algoritmo 5.9.

Exemplo 5.16 Para a matriz H do exemplo 5.14,

$$H = \begin{pmatrix} 0.987 & 0.905494 & -0.487 \\ -0.079 & 0.00281 & 0.479 \\ 0 & -0.030957 & 0.91519 \end{pmatrix}$$

o método Q-R demorou 6 iterações para atingir a matriz A_7 com o seguinte aspecto

$$\begin{pmatrix} 0.906232 & 0.002607 & 1.110422 \\ -0.002988 & 0.898785 & 0.483071 \\ 0 & -0.000028 & 0.099983 \end{pmatrix}.$$

Como a condição (5.27) não é verificada, para a matriz de 2×2 do canto superior esquerdo de A_7, o algoritmo vai resolver a equação (5.26), e obtêm-se dois valores reais $\lambda_1 = 0.904973$ e $\lambda_2 = 0.900044$. O outro valor próprio também é real e é visível na posição $(3,3)$ da matriz A_7. Assim, $\lambda_3 = 0.099983$.

Exemplo 5.17 Para a matriz H do exemplo 5.15

$$H = \begin{pmatrix} 32 & 17 & 1 \\ -2 & 31 & -4 \\ 0 & 27 & 18 \end{pmatrix},$$

o método Q-R demorou 206 iterações para atingir a matriz A_{207}

$$\begin{pmatrix} 26.412522 & 10.614886 & 31.068305 \\ -7.663244 & 27.587178 & -4.865120 \\ 0 & 0.000101 & 27.000300 \end{pmatrix}.$$

Como o elemento -7.663244 é 'diferente de zero', considera-se a matriz de 2×2 do canto superior esquerdo e resolve-se (5.26), dando origem aos seguintes valores próprios: $26.999850 + 8.999973\imath$ e $26.999850 - 8.999973\imath$. Resta o elemento 27.000300 que define um valor próprio real.

5.5.6 Cálculo do vector próprio associado a um valor complexo

Tal como já foi explicado em 5.4.8, deste Capítulo, o cálculo do vector próprio, x_i, associado ao valor λ_i $(i = 1, 2, \ldots, n)$ deve ser baseado no método iterativo da potência, iteração inversa e deslocada, por se tratar de um processo estável. Como a matriz não é simétrica, podemos ter valores e vectores próprios complexos. Suponha que a aproximação ao valor próprio é $\xi + \eta\,\imath$ e o correspondente vector próprio é $x + y\,\imath$, em que $\imath$ é a unidade imaginária. A iteração inversa e deslocada pode ser levada a cabo de várias maneiras. Uma delas, envolve a utilização de aritmética complexa na resolução de

$$[H - (\xi + \eta\,\imath)I]\bar{w}^{(k+1)} = \bar{v}^{(k)} \text{ com } \bar{v}^{(k)} = \frac{\bar{w}^{(k)}}{|\bar{w}^{(k)}|_\infty}, \text{ para } k = 1, 2, \ldots \tag{5.28}$$

sendo, agora, os vectores $\bar{v}$ e $\bar{w}$ definidos por

$$\bar{v} = v + t\,\imath, \quad \bar{w} = w + z\,\imath$$

e

$$v^{(k+1)} \to \text{ parte real}, x \quad e \quad t^{(k+1)} \to \text{ parte imaginária}, y \quad \text{quando} \quad k \to \infty.$$

O complexo $|\bar{w}^{(k)}|_\infty$ é o elemento do vector complexo $\bar{w}^{(k)}$ que tem maior módulo.

Pode-se, no entanto, implementar o processo iterativo em aritmética real, igualando as partes reais e as imaginárias do sistema de (5.28). Daqui resultam dois sistemas de n equações

$$\begin{aligned} (H - \xi I)w^{(k+1)} + \eta z^{(k+1)} &= v^{(k)}, \\ -\eta w^{(k+1)} + (H - \xi I)z^{(k+1)} &= t^{(k)} \end{aligned}$$

e, a partir deles, podemos obter outros dois equivalentes. Assim, multiplicando, à esquerda, o primeiro sistema por $(H-\xi I)$ e o segundo por $-\eta I$, e somando em seguida, obtém-se

$$[(H-\xi I)^2+\eta^2 I]w^{(k+1)} = (H-\xi I)v^{(k)} - \eta t^{(k)} \tag{5.29}$$

e, agora, multiplicando o primeiro sistema por ηI e o segundo por $(H-\xi I)$, e somando, tem-se

$$[(H-\xi I)^2+\eta^2 I]z^{(k+1)} = \eta v^{(k)} + (H-\xi I)t^{(k)}\,. \tag{5.30}$$

Os vectores, ambos de $\mathbf{R}^n$, $w^{(k+1)}$ (parte real de $\bar{w}$) e $z^{(k+1)}$ (parte imaginária de $\bar{w}$) podem ser determinados por um dos seguintes processos:

i) resolvendo primeiro a equação (5.29) em ordem a $w^{(k+1)}$ e substituindo este valor em

$$(H-\xi I)w^{(k+1)} + \eta z^{(k+1)} = v^{(k)}\,,$$

para determinar $z^{(k+1)}$;

ou

ii) resolvendo primeiro a equação (5.30) em ordem a $z^{(k+1)}$ e usando este valor em

$$-\eta w^{(k+1)} + (H-\xi I)z^{(k+1)} = t^{(k)}$$

para calcular $w^{(k+1)}$.

Se a estimativa $\xi+\eta\imath$ está muito próxima do valor exacto, λ_1 (complexo), a matriz $B(\lambda) = [(H-\xi I)^2+\eta^2 I]$ tem dois valores próprios muito próximos de zero e, em geral, $B(\lambda_1)$ tem característica $n-2$ uma vez que tem dois vectores próprios independentes $x+y\imath$ e $x-y\imath$. Este facto é revelado na decomposição triangular de $B(\lambda)$ com o aparecimento de dois elementos diagonais de U muito próximos de zero. O algoritmo 5.10 apresenta o método da potência, iteração inversa e deslocada para o cálculo de vectores próprios complexos associados a valores também complexos. A maior parte do algoritmo está implementado em aritmética real, embora no passo 11 haja necessidade de empregar alguns conhecimentos sobre operações com complexos. Considera-se como vector inicial do processo o vector unitário, $1+\imath$.

Algoritmo 5.10 :

1. `ler` n, n_{max}, ε_1, ε_2 `e para` $i=1,\ldots,n$ `ler` $(a_{il}, l=i-1,\ldots,n, l\neq 0)$
2. `ler` ξ `e` η `e fazer` $k=0$
3. `para` $i=1,\ldots,n$ `e` $l=1,\ldots,n$
   - 3.1. `se` $l=i$ `então fazer` $mat_{ii}=a_{ii}-\xi$
   - 3.2. `senão fazer` $mat_{il}=a_{il}$
4. `para` $i=1,\ldots,n$ `e` $k=1,\ldots,n$

4.1. calcular $b_{ik} = \sum_{l=1}^{n} mat_{il} mat_{lk}$

4.2. se $k = i$ calcular $b_{ik} = b_{ik} + \eta^2$

5. calcular os factores J e U da matriz $B = [(H - \xi I)^2 + \eta^2 I]$ tais que $JB = U$ (algoritmo 3.5, com selec="J e U")

6. se $|u_{nn}| < \varepsilon_2$ então fazer $u_{nn} = u_{n-1\,n-1} = \varepsilon_2$

7. para $i = 1, \ldots, n$ fazer $v_i^{(1)} = t_i^{(1)} = 1$

8. fazer $k = k + 1$

9. resolver o sistema $Uz^{(k+1)} = J[\eta v^{(k)} + (H - \xi I)t^{(k)}]$ em ordem a $z^{(k+1)}$:

   9.1. para $i = 1, \ldots, n$ fazer $vec_i = \eta v_i^{(k)} + \sum_{l=1}^{n} mat_{il} t_l^{(k)}$

   9.2. para $i = 1, \ldots, n$ calcular $aux_i = \sum_{l=1}^{n} j_{il} vec_l$

   9.3. calcular $z_n^{(k+1)} = \frac{aux_n}{u_{nn}}$

   9.4. para $i = n-1, \ldots, 1$ calcular $soma = \sum_{l=i+1}^{n} u_{il} z_l^{(k+1)}$ e $z_i^{(k+1)} = \frac{aux_i - soma}{u_{ii}}$

10. resolver $\eta w^{(k+1)} = -t^{(k)} + (H - \xi I)z^{(k+1)}$ em ordem a $w^{(k+1)}$:
    para $i = 1, \ldots, n$ calcular $w_i^{(k+1)} = \frac{1}{\eta}[-t_i^{(k)} + \sum_{l=1}^{n} mat_{il} z_l^{(k+1)}]$

11. determinar o elemento do vector complexo que tem maior módulo:
    para $i = 1, \ldots, n$ calcular el tal que $\sqrt{(w_{el}^{(k+1)})^2 + (z_{el}^{(k+1)})^2} \geq \sqrt{(w_i^{(k+1)})^2 + (z_i^{(k+1)})^2}$

12. calcular o vector normalizado:
    para $i = 1, \ldots, n$ calcular $v_i^{(k+1)} + t_i^{(k+1)}\,\imath = \frac{w_i^{(k+1)} + z_i^{(k+1)}\,\imath}{w_{el}^{(k+1)} + z_{el}^{(k+1)}\,\imath}$

13. calcular o vector resíduo, $r + s\,\imath$:
    para $i = 1, \ldots, n$ calcular $r_i = \sum_{l=1}^{n} a_{il} v_l^{(k+1)} - \xi v_i^{(k+1)} + \eta t_i^{(k+1)}$ e
    $s_i = \sum_{l=1}^{n} a_{il} t_l^{(k+1)} - \xi t_i^{(k+1)} - \eta v_i^{(k+1)}$

14. calcular $norma = \sqrt{\sum_{i=1}^{n} r_i^2 + s_i^2}$

15. se $norma > \varepsilon_1$ então

    15.1. se $k \geq n_{max}$ entã recomeçar, ir para o passo 2

    15.2. ir para o passo 8

16. terminar com $x(H) = x + y\,\imath \leftarrow (v_i^{(k+1)} + t_i^{(k+1)}\,\imath, i = 1, \ldots, n)$.

Finalmente, para se obterem os vectores próprios da matriz original A, $x_i(A)$, $i = 1, \ldots, n$, que correspondem aos vectores calculados da forma Hessenberg, $x_i(H)$, pelo processo acabado de descrever, basta multiplicá-los, à esquerda, pelas matrizes da transformação Hessenberg $M = I_{2,j} M_1 I_{3,k} M_2 \ldots I_{n-1,n} M_{n-2}$, com $j = 3, \ldots,$ ou n, $k = 4, \ldots,$ ou $n, \ldots,$

$$x_i(A) = M x_i(H)\,.$$

O algoritmo 5.8 fornece a matriz de transformação M.

5.5.7 Implementação. Rotina VECCOM

A rotina VECCOM implementa o algoritmo 5.10, que calcula, para cada valor próprio complexo, o vector próprio associado de H. Se os valores forem reais em vez da rotina VECCOM deve ser usada a VALVEC (algoritmo 5.4).

Os valores dados às tolerâncias são $\varepsilon_1 = 10^{-3}$ e $\varepsilon_2 = 10^{-4}$.

Exemplo 5.18 Para a matriz A do exemplo 5.15, obtém-se a forma Hessenberg

$$H = \begin{pmatrix} 32 & 17 & 1 \\ -2 & 31 & -4 \\ 0 & 27 & 18 \end{pmatrix},$$

e os seguintes valores próprios (exemplo 5.17):

$$\lambda_1 = 26.999850 + 8.999973\imath$$

$$\lambda_2 = 26.999850 - 8.999973\imath$$

$$\lambda_3 = 27.000300.$$

A rotina VECCOM fornece para o primeiro valor complexo, o vector associado de H,

$$x_1(H) = \begin{pmatrix} 0.166656 - 0.833320\imath \\ 0.333330 + 0.333335\imath \\ 1 \end{pmatrix}$$

e para o segundo valor complexo, surge o vector conjugado deste. Para o valor real, a rotina VALVEC fornece o vector $x_3(H) = (1, -0.25, -0.75)^T$, melhorando a aproximação do valor para 27.

Utilizando a matriz da transformação de A em H, dada pela rotina HESSEN

$$M = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -2 & 1 \end{pmatrix}$$

calcula-se o vector próprio de A: $x_1(A) = Mx_1(H)$

$$x_1(A) = \begin{pmatrix} 0.166656 - 0.833320\imath \\ 0.333330 + 0.333335\imath \\ 0.333340 - 0.666670\imath \end{pmatrix},$$

o vector $x_2(A) = Mx_2(H)$ é o conjugado de $x_1(A)$ e $x_3(A) = (1, -0.25, -0.25)^T = Mx_3(H)$.

5.6 Problemas

1. Para a matriz quadrada de ordem 3

$$A = \begin{pmatrix} 3 & -1 + 10^{-6} & 0 \\ -1 + 10^{-6} & 3 & 0 \\ 0 & 0 & 4 \end{pmatrix}$$

calcule o valor próprio de maior módulo e o vector próprio que lhe está associado. Use, para os parâmetros ε_1 e ε_2 do critério de paragem do processo iterativo, o valor 10^{-4}.

Face aos resultados obtidos, que conclusões pode tirar relativamente à proximidade deste valor próprio com os outros?

2. Use o método da potência, iteração inversa, para calcular o valor próprio de menor módulo da matriz

$$\begin{pmatrix} 0.4 & 0.0000002 \\ 0.0000001 & 0.4 \end{pmatrix}$$

3. Os valores próprios da matriz

$$A = \begin{pmatrix} 2 & 6 \\ -2 & -5 \end{pmatrix}$$

são reais. Calcule o vector próprio que está associado ao valor próprio mais próximo de -2.0011.

4. A matriz

$$A = \begin{pmatrix} -149 & -50 & -154 \\ 537 & 180 & 546 \\ -27 & -9 & -25 \end{pmatrix}$$

tem os seguintes valores próprios:

$$\lambda_1 = 3, \;\; \lambda_2 = 2 \;\text{ e }\; \lambda_3 = 1.$$

Se o elemento a_{22} for perturbado para 180.01, a matriz resultante passa a ter os seguintes valores:

$$\lambda_1 = 3.50189944, \;\; \lambda_2 = 2.30083490 \;\text{ e }\; \lambda_3 = 0.20726565.$$

Como explica estas diferenças?

5. Estude o condicionamento do vector próprio associado ao valor próprio de maior módulo 0.905, da matriz

$$\begin{pmatrix} 0.987 & 0.400 & -0.487 \\ -0.079 & 0.500 & 0.479 \\ 0.082 & 0.400 & 0.418 \end{pmatrix}.$$

Os outros valores próprios da matriz são:

$$\lambda_2 = 0.900 \;\text{ e }\; \lambda_3 = 0.100$$

6. Calcule o conjunto dos três valores próprios da matriz tridiagonal simétrica

$$A = \begin{pmatrix} 3 & 1 & 0 \\ 1 & 5 & 2 \\ 0 & 2 & 4 \end{pmatrix}$$

Verifique que dois dos valores próprios pertencem ao intervalo $[2, 6]$ e que o outro é maior do que 6.

7. Use a transformação de Householder para transformar a matriz simétrica de ordem 3

$$\begin{pmatrix} 3 & 1 & 4 \\ 1 & 7 & 2 \\ 4 & 2 & 0 \end{pmatrix}$$

à forma tridiagonal. Utilize a propriedade das sequências de Stürm para calcular os valores próprios desta matriz.

Um dos valores próprios é aproximadamente igual a 8.5. Use o método da potência para calcular o vector próprio que lhe está associado.

8. Calcule o par de valores próprios complexos conjugados da matriz

$$\begin{pmatrix} 4 & -1 \\ 2 & 2 \end{pmatrix}$$

usando os algoritmos 5.8 e 5.9.

Para os valores calculados implemente o algoritmo 5.10 para calcular os vectores próprios associados.

9. A matriz não simétrica de ordem 4

$$\begin{pmatrix} 4 & 3 & 2 & 1 \\ 5 & 6 & -7 & 8 \\ 0 & -2 & 1 & 9 \\ 0 & 0 & 3 & -4 \end{pmatrix}$$

está na forma quase triangular superior. Use o algoritmo 5.9 para calcular os valores próprios. Verifique que são todos reais, um negativo e os outros positivos.

Para o valor próprio mais próximo de zero, calcule o vector próprio a ele associado, usando o algoritmo 5.4

Capítulo 6

Interpolação polinomial

6.1 Introdução

6.1.1 Forma geral do problema

O problema da aproximação consiste em determinar uma *função aproximação* que descreva o melhor possível o comportamento de um conjunto de pontos $(x_0, f_0), (x_1, f_1), \ldots, (x_n, f_n)$. Este conjunto de pontos pode ter surgido de observações efectuadas ou medições feitas durante a realização de uma experiência e, neste caso, a função aproximação pode ser usada para prever valores em pontos intermédios (interpolar) ou para formular um modelo matemático que descreve o processo em causa. O conjunto de pontos também pode ser tomado como informação retirada de uma função com uma expressão complicada e analiticamente difícil de calcular, $f(x)$. Para este caso, uma função aproximação fácil e rápida de calcular pode ser útil para determinar (estimar) valores de f. Assim, de um modo geral a formulação deste tipo de problema é a seguinte:

- Dado um conjunto de pontos $(x_i, f_i), i = 0, \ldots, n$ e um conjunto de funções $\mathcal{P}$, determinar uma função aproximação $p \in \mathcal{P}$ que melhor descreve o comportamento dos dados, de acordo com uma certa *medida*.

Neste Capítulo consideram-se apenas funções de aproximação polinomiais. O conjunto $\mathcal{P}$ é o conjunto dos polinómios e p é um polinómio de um certo grau. A existência de um polinómio para aproximar uma função dada está garantida pelo Teorema de Weierstrass[1].

Teorema 6.1 (de aproximação de Weierstrass) : Sejam $-\infty < a < b < +\infty$ e f uma função arbitrária e contínua, $f \in C[a, b]$. Então, para todo o $\varepsilon > 0$, existe um $n \in N$ e um

[1]K. Weierstrass viveu entre 1815 e 1897. Os seus teoremas de aproximação foram estabelecidos em 1885, num trabalho editado em Berlim. Weierstrass não providenciou as provas construtivas dos teoremas, no entanto, em 1908, E. Landau e H. Lebesgue, e em 1912, S.N. Bernstein, forneceram provas alternativas. Weierstrass é considerado um dos fundadores da moderna Teoria das Funções (Hämmerlin e Hoffmann (1991)).

polinómio $p \in P_n$ *tal que*

$$\|f - p\| < \varepsilon,$$

em que $C[a,b]$ *é o espaço das funções contínuas no intervalo finito e fechado* $[a,b]$ *e* P_n *é o espaço linear de dimensão* $n+1$ *de todos os polinómios de grau máximo* n*, que tomam valores reais, definido por*

$$P_n = \{p \in C(-\infty, +\infty) \, : \, p_n(x) = \sum_{i=0}^{n} a_i x^i\}.$$

6.1.2 Características do problema

Uma das *medidas* usadas para determinar a proximidade de p em relação a f define as *condições de interpolação*

$$p(x_i) = f_i \,, \quad \text{para} \;\; i = 0, \ldots, n \tag{6.1}$$

sendo $f_i = f(x_i)$ e a aproximação chama-se *interpolação polinomial*. O polinómio passa por cada um dos pontos do conjunto e o vector resíduo $r(x) = f - p(x)$ é nulo em cada um desses pontos, uma vez que em cada um deles $r(x_i) = 0$. Vamos considerar duas classes diferentes $\mathcal{P}$ de polinómios interpoladores. Os polinómios normais, que são discutidos em 6.2. e 6.3. e os polinómios segmentados, também conhecidos por *splines*, que aparecem em 6.4.

Numa grande parte dos problemas, as condições de interpolação usadas como medida da proximidade não são adequadas. Por exemplo, se o número de pontos m for muito elevado, um polinómio interpolador a passar por todos esses pontos pode esconder um certa tendência monótona dos dados. Nesta situação justifica-se a construção de uma função aproximação que não passe necessariamente pelos pontos, mas antes, ajuste o melhor possível a mancha de pontos, no sentido dos mínimos quadrados, isto é, no sentido de que *a soma dos quadrados dos resíduos* seja mínima,

$$\min \sum_{i=1}^{m} [f_i - p(x_i)]^2 \,.$$

Pela importância desta formulação foi reservado o Capítulo 7 para estudar o ajuste de funções, polinomiais e não só, pela técnica dos mínimos quadrados.

Outra *medida* de interesse considera a norma infinita do vector resíduo,

$$\|r(x)\|_\infty = \max_{1 \le i \le m} |f_i - p(x_i)|$$

e tem como objectivo minimizá-la. O polinómio resultante desta condição chama-se aproximação minimax ou Chebyshev[2].

Existe alguma liberdade na escolha da classe de funções $\mathcal{P}$ e na *medida* da melhor proximidade entre f e p. Essa escolha deve ter em conta o tipo de dados que se tem e ser baseada em três critérios:

1. o custo da construção da função p,
2. o custo da determinação de valores de p,
3. a precisão.

O segundo critério é o mais importante. Pressupõe que há necessidade de calcular p, muitas vezes, e este custo vai de certeza dominar quando se compara com o custo da determinação do modelo p.

6.1.3 Índice de algoritmos

Neste Capítulo são apresentados quatro algoritmos. A lista é a seguinte:

Algoritmo 6.1 Interpolação directa a partir de uma tabela de valores, para pontos igualmente espaçados, com diferenças descendentes, ascendentes ou centrais

Algoritmo 6.2 Interpolação inversa a partir de uma tabela de valores, para pontos igualmente espaçados, com diferenças centrais

Algoritmo 6.3 Interpolação directa e inversa a partir de uma tabela de valores, para pontos não igualmente espaçados, com diferenças divididas

Algoritmo 6.4 Interpolação directa baseada em *splines* cúbicas naturais ou completas

Os algoritmos 6.1, 6.3 e 6.4 servem para interpolação directa a partir de um conjunto de valores de uma tabela. O primeiro pressupõe que os pontos da tabela estão igualmente espaçados, o segundo usa pontos com espaçamento não constante e o último constrói polinómios segmentados, isto é, polinómios de grau baixo, por exemplo de grau três, em segmentos de um intervalo $[a, b]$. Para o problema inverso, pode-se usar os algoritmos 6.2 ou 6.3 que geram polinómios interpoladores baseados em diferenças centrais, o primeiro, e divididas, o último.

[2]P. L. Chebyshev foi um matemático russo muito conhecido que viveu entre 1821 e 1894, tendo trabalhado quase toda a sua vida em S. Petersburgo. A sua influência é mais marcante nos domínios da Teoria dos números, Teoria das probabilidades e Teoria das funções ortogonais. As conhecidas funções de Chebyshev, $(1/2^{n-1})cos(n\,arccos(x))$ apareceram, em 1874, no seu artigo 'Sur les quadratures'. Em Moscovo e S.Petersburgo, durante o período de 1946 a 1951, foi feita uma compilação dos seus trabalhos, em 5 volumes: 'Works: Collected Works' (Goldstine (1977) e Hämmerlin e Hoffmann (1991)).

6.2 Interpolação com espaçamento constante

6.2.1 Introdução teórica. Erro de truncatura

Considere-se o caso em que os $n+1$ pontos $x_0, x_1, \ldots, x_{n-1}, x_n$ se encontram igualmente espaçados, isto é, conhecendo o primeiro ponto da lista, x_0, e o espaçamento entre eles, h, obtêm-se todos os outros da seguinte maneira:

$$x_j = x_0 + jh\,, \text{ para } j = 1, 2, \ldots, n.$$

Os correspondentes valores da função $f(x)$, nestes pontos, são:

$$f(x_0), f(x_1), f(x_2), \ldots, f(x_{n-1}), f(x_n),$$

que, para simplificar a notação passamos a usar f_j em lugar de $f(x_j)$, para $j = 0, 1, \ldots, n$.

O polinómio interpolador $p_n(x)$, de grau $\leq n$, baseado em $n+1$ pontos e na condição (6.1), pertence à classe dos *polinómios de colocação* e pode ser construído de várias maneiras. Em 6.2.2 obtém-se o polinómio interpolador de Gregory-Newton baseado em diferenças descendentes e em 6.2.3 o polinómio baseado nas diferenças ascendentes. Para pontos interpoladores no centro da tabela de valores é aconselhável o uso de um dos polinómios baseados em diferenças centrais, nomeadamente o de Stirling (veja-se em 6.2.4).

Dado um valor de $f(x)$ é possível calcular o valor do argumento que lhe corresponde, definindo o chamado *problema inverso*. Em 6.2.5 é deduzida uma fórmula baseada em diferenças centrais, que pode ser usada na interpolação inversa.

Vamos supor que todos os pontos x_j, $j = 0, 1, \ldots, n$ pertencem ao intervalo $[a, b]$. Da construção do polinómio interpolador $p_n(x)$, baseado nas condições (6.1), conclui-se que o erro $e_n(x) = f(x) - p_n(x)$ se anula nos $n+1$ pontos x_j. Interessa, agora, determinar limites do erro nos outros pontos do intervalo.

Teorema 6.2 : Se a derivada de ordem $n+1$ da função $f(x)$ é contínua no intervalo $[a, b]$ e todos os pontos $x_0, x_1, \ldots, x_n$ são distintos e pertencem a $[a, b]$, então a cada $x \in [a, b]$ corresponde um $\xi(x)$ tal que

$$e_n(x) = f(x) - p_n(x) = \pi_n(x)\frac{f^{(n+1)}(\xi(x))}{(n+1)!} \text{ com } a \leq \xi(x) \leq b, \tag{6.2}$$

sendo $\pi_n(x) = (x - x_0)(x - x_1)\ldots(x - x_{n-1})(x - x_n)$.

Um limite superior deste erro pode ser obtido se for possível calcular um majorante da derivada de ordem $n+1$ de f no intervalo $[a, b]$. Assim, se

$$|f^{(n+1)}(\xi)| \leq M_{n+1} \text{ para } \xi \in [a, b]$$

então um *limite superior do erro da aproximação polinomial* é

$$|e_n(x)| \leq \frac{M_{n+1}}{(n+1)!}|\pi_n(x)|.$$

6.2.2 Diferenças descendentes. Fórmula interpoladora de Gregory - Newton

Definem-se *diferenças descendentes de primeira ordem* como

$$\Delta f_j = f_{j+1} - f_j\,, \quad j = 0, \ldots, n-1\,,$$

as de *segunda ordem* como

$$\Delta^2 f_j = \Delta f_{j+1} - \Delta f_j = f_{j+2} - 2f_{j+1} + f_j\,, \quad j = 0, \ldots, n-2\,,$$

$$\ldots$$

e a *diferença descendente de ordem n* como

$$\Delta^n f_j = \Delta^{n-1} f_{j+1} - \Delta^{n-1} f_j = \sum_{r=0}^{n} (-1)^r \binom{n}{r} f_{j+n-r} \quad \text{para} \quad j = 0\,.$$

Estas diferenças podem ser colocadas numa tabela como a que a seguir se apresenta:

x_0	f_0					
$x \implies$		Δf_0				
x_1	f_1		$\Delta^2 f_0$			
		Δf_1		$\Delta^3 f_0$		
x_2	f_2		$\Delta^2 f_1$		$\Delta^4 f_0$	
		Δf_2		$\Delta^3 f_1$		$\Delta^n f_0$
$\ldots$	$\ldots$				$\ldots$	
x_{n-2}	f_{n-2}		$\Delta^2 f_{n-3}$		$\Delta^4 f_{n-4}$	
		Δf_{n-2}		$\Delta^3 f_{n-3}$		
x_{n-1}	f_{n-1}		$\Delta^2 f_{n-2}$			
		Δf_{n-1}				
x_n	f_n					

Tabela 6.1: Tabela das diferenças descendentes

Considerando a diferença Δ como um operador aplicado a f_j, então, é também possível definir outro operador, E, chamado *operador de uma etapa*, da seguinte maneira,

$$E f_j = f_{j+1}\,.$$

Como

$$\Delta f_j = E f_j - f_j \quad \text{ou} \quad (\Delta + 1) f_j = E f_j\,,$$

as potências de E, para k inteiro, podem ser calculadas a partir da relação

$$\begin{aligned} E^k f_j &= (1+\Delta)^k f_j \\ &= (1 + k\Delta + \binom{k}{2} \Delta^2 + \binom{k}{3} \Delta^3 + \cdots) f_j \end{aligned}$$

que originam a fórmula

$$f(x_j + kh) = f_j + k\Delta f_j + \binom{k}{2}\Delta^2 f_j + \binom{k}{3}\Delta^3 f_j + \cdots + \binom{k}{n}\Delta^n f_j + \cdots \qquad (6.3)$$

conhecida por *fórmula interpoladora de Gregory*[3]*-Newton com diferenças descendentes*. Um coeficiente da fórmula, por exemplo o $\binom{k}{3}$ é definido pela combinação de k, três a três, dada por $\frac{k!}{(k-3)!3!}$. Esta fórmula deve ser usada quando se pretende interpolar para pontos $x = x_j + kh$ colocados no início da tabela, como o que está indicado na tabela 6.1 com $x \implies$. Neste caso a fórmula (6.3) é usada fazendo $j = 0$ e $k = \frac{x-x_0}{h}$. O ponto x_j é aquele que se encontra imediatamente antes da posição do ponto interpolador x.

As manipulações feitas não podem ser justificadas apenas a partir das propriedades algébricas dos operadores Δ e E. Pressupõe-se também que a série infinita converge e tem soma igual a $f(x_j + kh)$. Isto é verdadeiro para um número bem limitado de casos onde se impõem grandes restrições na função f. No entanto, o que mais interessa é o *erro de truncatura* cometido quando a série (6.3) é truncada após o termo em função da diferença de ordem n. Nestas condições, a função está a ser aproximada por um polinómio de grau n e escreve-se

$$f(x_j + kh) = p_n(k) + e_n(k)$$

em que

$$p_n(k) = f_j + k\Delta f_j + \binom{k}{2}\Delta^2 f_j + \cdots + \binom{k}{n}\Delta^n f_j.$$

O $p_n(k)$ é um polinómio em k de grau n. É possível provar que $p_n(k) = f(x_j + kh)$ para $k = 0, 1, \ldots, n$ (nos pontos interpoladores) e o erro desta aproximação polinomial, que já foi estabelecido em 6.2.1., é dado por

$$e_n(k) = \frac{k(k-1)\ldots(k-n)}{(n+1)!}\left[\frac{d}{dk}^{(n+1)} f(x_j + kh)\right]_{k=\theta} \qquad (6.4)$$

$$= \frac{k(k-1)\ldots(k-n)}{(n+1)!} h^{n+1} f^{(n+1)}(x_j + \theta h) \qquad (6.5)$$

para algum valor de θ do intervalo que contém $0, 1, \ldots, n$ e k.

[3]Esta fórmula interpoladora foi descoberta por J. Gregory e I. Newton independentemente um do outro. Tal como Newton, o interesse de Gregory pela interpolação está relacionado com a integração numérica. Em 1668 Gregory mostrou a natureza recíproca da diferenciação e da integração, no trabalho 'Geometricae pars universalis, inserviens quantitatum curvarum transmutationi et mensurae'. Em 'Exercitationes Geometricae' (1668), Gregory publicou a regra de Simpson de integração, embora ela tenha sido encontrada na sua forma geométrica, anos antes, em 1639, por Cavalieri. A própria fórmula de integração de Gregory apareceu discutida numa carta endereçada a Collins, datada de 1670 (ver o Capítulo 8 sobre Integração Numérica). Gregory era escocês, nasceu em 1638, foi professor de matemática do Colégio de St. Andrews, em Oxford, até 1674, e da Universidade de Edimburgo até à sua morte em 1675 (Goldstine (1977) e Hämmerlin e Hoffmann (1991)).

6.2.3 Diferenças ascendentes. Fórmula interpoladora de Gregory - Newton

As *diferenças ascendentes* definem-se da seguinte maneira:

$$\nabla f_j = f_j - f_{j-1}\,, \quad j = 1, \ldots, n$$

para as de *primeira ordem*,

$$\nabla^2 f_j = \nabla f_j - \nabla f_{j-1} = f_j - 2f_{j-1} + f_{j-2}\,, \quad j = 2, \ldots, n$$

para as de *segunda ordem*,

$$\cdots$$

$$\nabla^n f_j = \nabla^{n-1} f_j - \nabla^{n-1} f_{j-1} = \sum_{r=0}^{n} (-1)^r \binom{n}{r} f_{j-r} \quad \text{para} \quad j = n$$

para a de *ordem* n. A tabela que corresponde às diferenças ascendentes de uma função $f(x)$, conhecida para um conjunto de $n+1$ pontos, é a seguinte:

x_0	f_0					
		∇f_1				
x_1	f_1		$\nabla^2 f_2$			
		∇f_2		$\nabla^3 f_3$		
x_2	f_2		$\nabla^2 f_3$		$\nabla^4 f_4$	
		∇f_3		$\nabla^3 f_4$		$\nabla^n f_n$
$\cdots$	$\cdots$				$\cdots$	
x_{n-2}	f_{n-2}		$\nabla^2 f_{n-1}$		$\nabla^4 f_n$	
		∇f_{n-1}		$\nabla^3 f_n$		
x_{n-1}	f_{n-1}		$\nabla^2 f_n$			
$x \implies$		∇f_n				
x_n	f_n					

Tabela 6.2: Tabela das diferenças ascendentes

A *fórmula interpoladora de Gregory-Newton com diferenças ascendentes* é obtida a partir da relação

$$\nabla f_j = f_j - E^{-1} f_j \iff (1-\nabla) f_j = E^{-1} f_j \iff (1-\nabla)^{-1} f_j = E f_j\,.$$

Assim,

$$\begin{aligned} E^k f_j &= (1-\nabla)^{-k} f_j \\ &= (1 + k\nabla + \frac{k(k+1)}{2!}\nabla^2 + \frac{k(k+1)(k+2)}{3!}\nabla^3 + \cdots) f_j \end{aligned}$$

e

$$f(x_j + kh) = f_j + k\nabla f_j + \binom{k+1}{2} \nabla^2 f_j + \binom{k+2}{3} \nabla^3 f_j + \cdots + \binom{k+n-1}{n} \nabla^n f_j + \cdots \quad (6.6)$$

Um coeficiente da fórmula, por exemplo o $\binom{k+1}{2}$ é definido pela combinação de $k+1$, dois a dois, dada por $\frac{(k+1)!}{(k-1)!2!}$.

Truncando a série depois do termo que envolve ∇^n, obtém-se uma aproximação polinomial de grau n, $p_n(x)$, a $f(x)$, e o resto ou erro de truncatura é dado por

$$e_n(k) = \frac{k(k+1)\ldots(k+n)}{(n+1)!} h^{n+1} f^{(n+1)}(\xi) \quad (6.7)$$

sendo ξ um ponto do intervalo definido pelos pontos usados para construir o polinómio interpolador.

A utilização desta fórmula para interpolação directa deve ser feita para pontos, $x = x_j + kh$, que se encontrem no fim da tabela, como o que está indicado na tabela 6.2 com $x \Longrightarrow$. Para este caso toma-se $j = n$ e $k = \frac{x - x_n}{h}$. Considera-se o ponto x_j como sendo o ponto colocado imediatamente a seguir a x. Esta é mais uma maneira de chegar ao polinómio (único) interpolador que passa pelos $n+1$ pontos $x_0, \ldots, x_n$.

6.2.4 Diferenças centrais. Fórmula interpoladora de Stirling

A melhor escolha para x_j é aquela que coloca o ponto interpolador, $x = x_j + kh$, tão perto quanto possível do centro do conjunto dos $n+1$ pontos usados na interpolação, pois nesta situação temos informação da função de um e de outro lado do ponto. Este objectivo é atingido, calculando as diferenças centrais de f e usando fórmulas interpoladoras construídas a partir destas diferenças.

As *diferenças centrais* são definidas por

$$\delta f_{j+\frac{1}{2}} = f_{j+1} - f_j\,, \quad j = 0, \ldots, n-1$$

para as de *primeira ordem*,

$$\delta^2 f_j = \delta f_{j+\frac{1}{2}} - \delta f_{j-\frac{1}{2}} = f_{j+1} - 2f_j + f_{j-1}\,, \quad j = 1, \ldots, n-1$$

para as de *segunda ordem*, e assim sucessivamente. De um modo geral, para as de *ordem impar* tem-se

$$\delta^{2l+1} f_{j+\frac{1}{2}} = \delta^{2l} f_{j+1} - \delta^{2l} f_j$$

e para as de *ordem par*,

$$\delta^{2l} f_j = \delta^{2l-1} f_{j+\frac{1}{2}} - \delta^{2l-1} f_{j-\frac{1}{2}}\,,$$

para qualquer l.

A tabela com as diferenças centrais da função f é a seguinte:

x_0	f_0						
		$\delta f_{\frac{1}{2}}$					
x_1	f_1		$\delta^2 f_1$				
		$\delta f_{\frac{3}{2}}$		$\delta^3 f_{\frac{3}{2}}$			
$\cdots$	$\cdots$			$\cdots$			
x_j	f_j		$\delta^2 f_j$		$\delta^4 f_j$	$\cdots$	
$x \implies$		$\delta f_{j+\frac{1}{2}}$		$\delta^3 f_{j+\frac{1}{2}}$			$\delta^n f_{j+\frac{1}{2}}$
x_{j+1}	f_{j+1}		$\delta^2 f_{j+1}$		$\delta^4 f_{j+1}$	$\cdots$	
$\cdots$	$\cdots$			$\cdots$			
		$\delta f_{n-\frac{3}{2}}$		$\delta^3 f_{n-\frac{3}{2}}$			
x_{n-1}	f_{n-1}		$\delta^2 f_{n-1}$				
		$\delta f_{n-\frac{1}{2}}$					
x_n	f_n						

Tabela 6.3: Tabela das diferenças centrais

Através da relação

$$E^k f_j = (1 + E^{\frac{1}{2}}\delta)^k f_j$$

e expandindo o binómio do lado direito, obtém-se uma fórmula com diferenças centrais conhecida por fórmula interpoladora descendente de Gauss,

$$E^k f_j = (1 + kE^{\frac{1}{2}}\delta + \frac{k(k-1)}{2!}E\delta^2 + \cdots)f_j$$

que pode ser escrita na forma

$$f(x_j + kh) = f_j + \binom{k}{1}\delta f_{j+\frac{1}{2}} + \binom{k}{2}\delta^2 f_j + \binom{k+1}{3}\delta^3 f_{j+\frac{1}{2}} + \binom{k+1}{4}\delta^4 f_j + \cdots$$

uma vez que $E^{\frac{1}{2}}\delta f_j = \delta f_{j+\frac{1}{2}}$ e $E\delta^2 f_j = (\delta^2 + E^{\frac{1}{2}}\delta^3)f_j$.

Usando um processo idêntico, mas agora com a relação

$$E^k f_j = (1 - E^{\frac{1}{2}}\delta)^{-k}$$

chega-se a

$$f(x_j + kh) = f_j + \binom{k}{1}\delta f_{j-\frac{1}{2}} + \binom{k+1}{2}\delta^2 f_j + \binom{k+1}{3}\delta^3 f_{j-\frac{1}{2}} + \binom{k+2}{4}\delta^4 f_j + \binom{k+2}{5}\delta^5 f_{j-\frac{1}{2}} + \cdots$$

e que é a fórmula interpoladora ascendente de Gauss.

Calculando a média aritmética das duas fórmulas de Gauss e definindo *as diferenças médias de ordem impar* por

$$\mu\delta f_j = \frac{1}{2}(\delta f_{j+\frac{1}{2}} + \delta f_{j-\frac{1}{2}}),$$

$$\mu\delta^3 f_j = \frac{1}{2}(\delta^3 f_{j+\frac{1}{2}} + \delta^3 f_{j-\frac{1}{2}}),$$
$$\dots$$

obtém-se a seguinte fórmula

$$f(x_j + kh) = f_j + \binom{k}{1}\mu\delta f_j + \frac{k}{2}\binom{k}{1}\delta^2 f_j + \binom{k+1}{3}\mu\delta^3 f_j +$$
$$+\frac{k}{4}\binom{k+1}{3}\delta^4 f_j + \binom{k+2}{5}\mu\delta^5 f_j + \frac{k}{6}\binom{k+2}{5}\delta^6 f_j + \cdots \tag{6.8}$$

que é conhecida por *fórmula interpoladora de Stirling*[4] *com diferenças centrais*. Se a expansão termina após o termo que envolve a diferença $\delta^{2n} f_j$, o polinómio truncado é de grau $2n$ e pode-se mostrar que toma os mesmos valores que a função $f(x)$ nos pontos $x_{j-n}, x_{j-(n-1)}, \dots, x_{j-1}, x_j,\ \ x_{j+1}, \dots, x_{j+(n-1)}, x_{j+n}$. O resto ou erro de truncatura é equivalente ao obtido pela fórmula do Teorema 6.2 baseado nos pontos interpoladores $x_{j-n}, \dots, x_j,\ \dots, x_{j+n}$ e pode ser escrito na forma

$$e_{2n}(k) = \frac{k(k^2-1)\dots(k^2-n^2)}{(2n+1)!}h^{2n+1} f^{(2n+1)}(\xi) \tag{6.9}$$

com ξ um ponto do intervalo definido pelos pontos usados na interpolação.

O algoritmo 6.1 calcula as diferenças da função $f(x)$ e o valor do polinómio interpolador, para um certo argumento *valor*, usando a fórmula interpoladora de Newton-Gregory com diferenças descendentes, no passo 4 do algoritmo, com diferenças ascendentes, no passo 5.1, ou a fórmula de Stirling com diferenças centrais, no passo 5.2. A selecção deve ser feita tendo como base a posição do ponto interpolador *valor*, em relação aos $n+1$ pontos da tabela. O parâmetro *ponto* do algoritmo deve indicar essa posição ('inicio', 'meio' ou 'fim').

Algoritmo 6.1 :

1. ler n, *grau* ($\leq n$), *ponto* ('inicio', 'meio ' ou 'fim ') e para $i = 0, 1, \dots, n$ ler x_i e $f(x_i)$
2. ler *valor* e calcular $h = x_1 - x_0$
3. para $i = 0, \dots, n$ fazer $d_{i0} = f(x_i)$
4. se *ponto* = 'inicio' então
   - 4.1. para $j = 1, \dots, n$ e $i = 0, \dots, n-j$ calcular $d_{ij} = d_{i+1\,j-1} - d_{i\,j-1}$
   - 4.2. calcular $k = \frac{valor - x_0}{h}$

[4] J. Stirling nasceu em 1692 na Escócia. Como estudante passou por Glasgow, Oxford e Veneza, tendo-se tornado amigo de Newton após o seu regresso a Inglaterra. Todo o seu trabalho foi influenciado pelas ideias de Newton. O seu trabalho mais importante 'Methodus Differentialis' foi publicado em 1730, traduzido para inglês em 1749, e nele aparecem os cálculos relativos aos famosos números de Stirling de primeira e segunda espécie. Morreu no ano de 1770 (Goldstine (1977)).

4.3. fazer $coef_0 = 1$ e para $j = 1, \ldots, grau$ calcular $coef_j = coef_{j-1}\frac{k-j+1}{j}$

4.4. calcular $polin = \sum_{j=0}^{grau} coef_j d_{0j}$

5. senão

5.1. se $ponto =$ 'fim ' então

5.1.1. para $j = 1, \ldots, n$ e $i = j, \ldots, n$ calcular $d_{ij} = d_{i\,j-1} - d_{i-1\,j-1}$

5.1.2. calcular $k = \frac{valor - x_n}{h}$

5.1.3. fazer $coef_0 = 1$ e para $j = 1, \ldots, grau$ calcular $coef_j = coef_{j-1}\frac{k+j-1}{j}$

5.1.4. calcular $polin = \sum_{j=0}^{grau} coef_j d_{nj}$

5.2. senão (para um ponto no meio da tabela)

5.2.1. para $j = 1, \ldots, n$ e $i = 0, \ldots, n-j$ calcular $d_{ij} = d_{i+1\,j-1} - d_{i\,j-1}$

5.2.2. para $i = 0, \ldots, n$ se $valor < x_i$ então fazer $m = i - 1$ e ir para o passo 5.2.3.

5.2.3. calcular $k = \frac{valor - x_m}{h}$

5.2.4. fazer $coef_0 = 1$, $coef_1 = k$, $p = m$, $num = k^2$ e $polin = d_{m0} + \frac{1}{2}(d_{p\,1} + d_{p-1\,1})coef_1$

5.2.5. para $j = 2, \ldots, grau$

5.2.5.1. se $j = n^o$ par então calcular $coef_j = coef_{j-2}\frac{num}{(j-1)j}$, fazer $p = p-1$ e calcular $polin = polin + coef_j d_{p\,j}$

5.2.5.2. senão calcular $num = k^2 - (\frac{j-1}{2})^2$, $coef_j = coef_{j-2}\frac{num}{(j-1)j}$ e $polin = polin + \frac{1}{2}(d_{p\,j} + d_{p-1\,j})coef_j$

6. terminar com $p(valor) \leftarrow polin$.

6.2.5 Implementação. Rotina DIFDIR

A rotina DIFDIR implementa o algoritmo 6.1 e os resultados obtidos para os dois exemplos que se seguem são:

Exemplo 6.1 Dada a tabela de valores de uma função $f(x)$

x_i	6.0	6.1	6.2	6.3	6.4	6.5	6.6
f_i	0.166670	0.163930	0.161290	0.158730	0.156250	0.153850	0.151520

com sete pontos, o polinómio interpolador de Gregory-Newton, de grau 2, que aproxima a função, fornece para o ponto interpolador $x = 6.52$, no fim da tabela, o valor 0.153378. Foram usadas diferenças ascendentes. A correspondente tabela das diferenças (desde as de primeira ordem à de sexta ordem) é a seguinte:

-0.002740					
	0.000100				
-0.002640		-0.000020			
	0.000080		0.000020		
-0.002560		-0.000000		-0.000020	
	0.000080		0.000000		0.000010
-0.002480		0.000000		-0.000010	
	0.000080		-0.000010		
-0.002400		-0.000010			
	0.000070				
-0.002330					

Para calcular uma estimativa do erro de truncatura cometido, considera-se $n = 2$ nas fórmulas (6.6) e (6.7). O polinómio interpolador corresponde a $p_2(x) = f_j + k\nabla f_j + \binom{k+1}{2} \nabla^2 f_j$ e

$$|e_2(k)| \leq \frac{k(k+1)(k+2)}{3!} h^3 M_3$$

sendo M_3 o majorante da derivada de ordem três da função $f(x)$, no intervalo definido pelos pontos usados na interpolação. Como não se conhece a expressão de f, $h^3 M_3$ é aproximado pela diferença de ordem três de f mais perto da região, em valor absoluto, e que é 0.000010. Como $k = \frac{6.52-6.6}{0.1} = -0.8$, tem-se

$$|e_2(6.52)| \leq (0.032)(0.000010) = 3.2 \times 10^{-7}.$$

Exemplo 6.2 Dada a tabela de $f(x)$

x_i	f_i
5.0	0.0639269
5.1	0.0800431
5.2	0.0987853
5.3	0.1202508
5.4	0.1444762
5.5	0.1714318
5.6	0.2010186
5.7	0.2330688
5.8	0.2673489
5.9	0.3035673
6.0	0.3413825

com onze pontos, o polinómio interpolador de Stirling de grau 4 dá o valor 0.157617 como aproximação a $f(5.45)$.

6.2.6 Cálculo da derivada para pontos da tabela

Depois de calcular uma aproximação polinomial a uma função dada, $f(x)$, torna-se simples o cálculo de uma aproximação à sua derivada ou ao integral. O problema da integração

será tratado no Capítulo 8. O problema da diferenciação será estudado no Capítulo 9. No entanto, e de uma forma muito sucinta pode-se mostrar como as fórmulas de interpolação podem ser usadas para *aproximar a derivada, $f'(x)$, em pontos da tabela* $x = x_j$, $j = 0, \ldots, n$. Considere-se, por exemplo, a fórmula interpoladora de Gregory-Newton com diferenças descendentes apresentada em (6.3),

$$f(x_j + kh) = f_j + k\Delta f_j + \frac{k(k-1)}{2}\Delta^2 f_j + \frac{k(k-1)(k-2)}{6}\Delta^3 f_j + \cdots$$

e determine-se $f'(x_j)$,

$$\begin{aligned} f'(x_j) &= [\frac{d\, f(x_j + kh)}{dx}]_{k=0} \\ &= \frac{1}{h}[\frac{d\, f(x_j + kh)}{dk}]_{k=0} \\ &= \frac{1}{h}[\Delta f_j - \frac{1}{2}\Delta^2 f_j + \frac{1}{3}\Delta^3 f_j - \cdots] \end{aligned}$$

sendo $k = \frac{x - x_j}{h}$.

O resto, ou erro de truncatura da aproximação obtém-se pelo mesmo processo, a partir do termo resto da fórmula de interpolação. Assim, se a aproximação à derivada inclui os termos até à diferença de ordem n, usando o resultado do Teorema 6.2, tem-se que o resto desta aproximação diferencial é

$$\begin{aligned} e'_n(x) &= \frac{d}{dx}[\pi_n(x)\frac{f^{(n+1)}(\xi(x))}{(n+1)!}] \\ &= \frac{1}{(n+1)!}[\pi'_n(x) f^{(n+1)}(\xi(x)) + \pi_n(x)\frac{d}{dx} f^{(n+1)}(\xi(x))] \end{aligned}$$

e

$$e'_n(x_j) = \pi'_n(x_j)\frac{f^{(n+1)}(\xi(x))}{(n+1)!}$$

uma vez que $\pi_n(x_j) = 0$.

6.2.7 Interpolação inversa. Fórmula interpoladora de Everett

O problema da *interpolação inversa* consiste em determinar, a partir de uma tabela de valores de uma função, $f(x)$, o valor do argumento que corresponde a um dado valor de $f(x)$.

Para resolver um problema de interpolação inversa usa-se uma técnica de aproximações sucessivas baseada numa das fórmulas interpoladoras com diferenças centrais. Por exemplo, pode-se usar uma fórmula interpoladora onde só intervêm diferenças centrais de ordem par. A sua dedução é feita a partir da fórmula descendente de Gauss escrita na forma

$$\begin{aligned} f(x_j + kh) &= (f_j + k\delta f_{j+\frac{1}{2}}) + \frac{k(k-1)}{2!}(\delta^2 f_j + \frac{k+1}{3}\delta^3 f_{j+\frac{1}{2}}) \\ &+ \frac{k(k^2-1)(k-2)}{4!}(\delta^4 f_j + \frac{k+2}{5}\delta^5 f_{j+\frac{1}{2}}) + \cdots \end{aligned}$$

e da relação

$$\delta^{2r+1} f_{j+\frac{1}{2}} = \delta^{2r} f_{j+1} - \delta^{2r} f_j.$$

Substituindo esta relação em cada uma das diferenças de ordem impar que aparecem na fórmula de Gauss, obtém-se a *fórmula interpoladora de Everett com diferenças centrais* de ordem par

$$f(x_j + kh) = (1-k) f_j + k f_{j+1} + E_2 \delta^2 f_j + E_3 \delta^2 f_{j+1} + E_4 \delta^4 f_j + E_5 \delta^4 f_{j+1} + \cdots \quad (6.10)$$

em que os coeficientes E_i, $i = 2, 3, \ldots$ são dados por

$$E_2 = -\frac{k(k-1)(k-2)}{3!}, \quad E_3 = \frac{(k+1)k(k-1)}{3!},$$

$$E_4 = -\frac{(k+1)k(k-1)(k-2)(k-3)}{5!}, \quad E_5 = \frac{(k+2)(k+1)k(k-1)(k-2)}{5!}, \quad \ldots$$

A fórmula (6.10) pode ser rearranjada

$$k = \frac{1}{f_{j+1} - f_j}[f(x_j + kh) - f_j - E_2 \delta^2 f_j - E_3 \delta^2 f_{j+1} - E_4 \delta^4 f_j - E_5 \delta^4 f_{j+1} - \cdots] \quad (6.11)$$

e resolvida em ordem a k por aproximações sucessivas, tomando para primeira aproximação do processo, o valor

$$k^{(1)} = \frac{f(x) - f_j}{f_{j+1} - f_j}$$

uma vez que é conhecido $f(x) = f(x_j + kh)$. Obtém-se, assim, iterativamente uma sequência de valores para k. Em primeiro lugar os coeficientes $E_2, E_3, E_4, E_5, \ldots$ são calculados em função de $k^{(1)}$ e substituídos na fórmula (6.11) para calcular o $k^{(2)}$. Com este valor, repete-se o processo do cálculo dos $E's$ e posteriormente de $k^{(3)}$. O processo deve terminar quando os valores de k estabilizarem e convergirem para $\bar{k}$. Este valor será usado na relação $x = x_j + \bar{k}h$ para se obter o argumento x pretendido. Na fórmula (6.11) usam-se tantos termos quantas as diferenças disponíveis na tabela. Se a partir de uma certa ordem as diferenças tiverem valores médios próximos de zero, os termos correspondentes na fórmula (6.11) podem ser ignorados.

No algoritmo 6.2 é apresentado este processo de aproximações sucessivas baseado na fórmula de Everett com diferenças centrais.

Algoritmo 6.2 :

1. `ler` n, n_{max}, ε_1, ε_2, `calcular` tol `e para` $i = 0, 1, \ldots, n$ `ler` x_i `e` $f(x_i)$
2. `ler` $valor$, `calcular` $h = x_1 - x_0$ `e fazer` $ordem = n$
3. `para` $i = 0, \ldots, n$ `fazer` $d_{i0} = f(x_i)$
4. `para` $j = 1, \ldots, n$

   4.1. `para` $i = 0, \ldots, n - j$ `calcular` $d_{ij} = d_{i+1\,j-1} - d_{i\,j-1}$

4.2. calcular $media = \frac{\sum_{i=0}^{n-j} d_{ij}}{n-j+1}$

4.3. se $|media| < \varepsilon_1$ então fazer $ordem = j-1$ e ir para o passo 5

5. 5.1. se $f(x_0) \leq valor$ então para $i = 1, \ldots, n$, se $valor < f(x_i)$ então fazer $m = i-1$ e ir para o passo 6

5.2. senão para $i = 1, \ldots, n$, se $valor > f(x_i)$ então fazer $m = i-1$ e ir para o passo 6

6. fazer $l = 0$ e calcular $k^{(1)} = \frac{valor - d_{m0}}{d_{m+1\,0} - d_{m0}}$

7. se $ordem = n^{o.}\, par$ então

7.1. se $ordem = n$ então $ordem = ordem - 1$

7.2. senão $ordem = ordem + 1$

8. fazer $l = l+1$, $iter = valor - d_{m0}$, fazer $p = m$, $coef_0 = -(k^{(l)} - 1)$, $coef_1 = k^{(l)}$ e $termo = k^{(l)}$

9. para $j = 2, \ldots, ordem$ (calcular os E's em função de $k^{(l)}$):

9.1. se $j = n^{o.}\, par$ então calcular $coef_j = coef_{j-2}\, termo\, \frac{(k^{(l)} - \frac{j}{2} - 1)}{j(j+1)}$ e $iter = iter - d_{p-1\,j}\, coef_j$

9.2. senão calcular $termo = (k^{(l)} + \frac{j-1}{2})$, $coef_j = coef_{j-2}\, termo\, \frac{(k^{(l)} - \frac{j-1}{2})}{(j-1)j}$, $iter = iter - d_{p\,j-1} coef_j$ e fazer $p = p - 1$

10. calcular $k^{(l+1)} = \frac{iter}{d_{m+1\,0} - d_{m0}}$

11. se $|k^{(l+1)}| \leq tol$ então terminar, o processo convergiu para zero ($x \leftarrow x_m$)

12. se $\frac{|k^{(l+1)} - k^{(l)}|}{|k^{(l+1)}|} \leq \varepsilon_2$ então ir para o passo 15

13. se $l \geq n_{max}$ então terminar, o processo não está a convergir

14. ir para o passo 8

15. calcular $argum = x_m + k^{(l+1)} h$

16. terminar com $x \leftarrow argum$.

6.2.8 Implementação. Rotina DIFINV

A rotina DIFINV surge da implementação do algoritmo 6.2. Aos parâmetros ε_1 e ε_2 são dados respectivamente os seguintes valores: 10^{-3} e 10^{-4}.

Exemplo 6.3 Para calcular o zero da função dada pela seguinte tabela de quatro pontos

x_i	1.2	1.3	1.4	1.5
f_i	0.472	0.103	-0.344	-0.875

a rotina DIFINV calcula os seguintes valores de k : $0.230425, 0.246385, 0.247143, 0.247178$ e 0.247179 donde se tira $x = 1.324718$.

Exemplo 6.4 O valor de x para o qual $f(x) = 2$, para a função dada pela tabela

x_i	0.0	0.2	0.4	0.6	0.8	1.0	1.2	1.4
f_i	1.000	1.221	1.492	1.822	2.226	2.718	3.320	4.056

com oito pontos, é 0.693057. Este valor foi obtido ao fim de três iterações calculadas pela fórmula (6.11). Para $k^{(1)} = 0.440594$, os restantes valores de k foram: $0.465024, 0.465285$ e 0.465287.

6.3 Interpolação com espaçamento não constante

6.3.1 Diferenças divididas

Considere-se agora a função $f(x)$ tabelada para um conjunto de $n+1$ pontos $x_0, x_1, x_2, \ldots, x_{n-1}, x_n$ não igualmente espaçados. Os correspondentes valores da função são designados respectivamente por $f_0, f_1, f_2, \ldots, f_{n-1}, f_n$. Define-se *diferença dividida de primeira ordem* relativa a x_j e a x_{j+1} por

$$[x_j, x_{j+1}] = \frac{f_j - f_{j+1}}{x_j - x_{j+1}}$$

para $j = 0, 1, \ldots, n-1$. Se duas diferenças divididas de primeira ordem têm um argumento comum, então podemos definir a *diferença dividida de segunda ordem* por

$$[x_j, x_{j+1}, x_{j+2}] = \frac{[x_j, x_{j+1}] - [x_{j+1}, x_{j+2}]}{x_j - x_{j+2}}, \quad \text{para} \quad j = 0, \ldots, n-2$$

e, assim sucessivamente. Define-se *diferença dividida de ordem n* por

$$[x_0, x_1, \ldots, x_{n-1}, x_n] = \frac{[x_0, x_1, \ldots, x_{n-1}] - [x_1, x_2, \ldots, x_n]}{x_0 - x_n}.$$

A tabela das diferenças que corresponde às definições apresentadas é a seguinte:

x_0	f_0				
		$[x_0, x_1]$			
x_1	f_1		$[x_0, x_1, x_2]$		
		$[x_1, x_2]$		$[x_0, x_1, x_2, x_3]$	
x_2	f_2		$[x_1, x_2, x_3]$		
		$[x_2, x_3]$		$[x_1, x_2, x_3, x_4]$	$[x_0, x_1, \ldots, x_{n-1}, x_n]$
...	...				
x_{n-2}	f_{n-2}		$[x_{n-3}, x_{n-2}, x_{n-1}]$		
		$[x_{n-2}, x_{n-1}]$		$[x_{n-3}, x_{n-2}, x_{n-1}, x_n]$	
x_{n-1}	f_{n-1}		$[x_{n-2}, x_{n-1}, x_n]$		
		$[x_{n-1}, x_n]$			
x_n	f_n				

Tabela 6.4: Tabela das diferenças divididas de f

As diferenças divididas gozam das seguintes propriedades:

1. As diferenças divididas são funções simétricas dos seus argumentos, isto é, o valor da diferença é independente da ordem por que se consideram os seus argumentos;
2. Se uma função $f_j = u_j + v_j$ para um conjunto de valores do argumento $x = x_j$, $(j = 0, \ldots, n)$, então qualquer diferença dividida de $f(x)$, formada a partir destes valores de x_j, é igual à soma das correspondentes diferenças divididas de $u(x)$ e $v(x)$;
3. Uma diferença dividida de $cf(x)$, sendo c uma constante, é igual ao produto de c pela correspondente diferença dividida de $f(x)$;
4. As diferenças divididas de ordem n da função x^n são iguais a 1 e, as de ordem $r > n$ são nulas;

   e consequentemente,

5. As diferenças divididas de ordem n de um polinómio de grau n são iguais e diferentes de zero.

6.3.2 Fórmula interpoladora de Newton. Erro de truncatura

Das definições de diferenças divididas apresentadas pode-se concluir que

$$f(x) = f_0 + (x - x_0)[x_0, x]$$

$$[x_0, x] = [x_0, x_1] + (x - x_1)[x_0, x_1, x]$$

$$[x_0, x_1, x] = [x_0, x_1, x_2] + (x - x_2)[x_0, x_1, x_2, x]$$

$$\ldots$$

$$[x_0, x_1, \ldots, x_{n-1}, x] = [x_0, x_1, \ldots, x_n] + (x - x_n)[x_0, x_1, \ldots, x_{n-1}, x_n, x]$$

$$\ldots$$

e por substituição sucessiva, chega-se à fórmula

$$f(x_j + kh) = f_0 + (x - x_0)[x_0, x_1] + (x - x_0)(x - x_1)[x_0, x_1, x_2] + \tag{6.12}$$

$$+(x - x_0)(x - x_1)(x - x_2)[x_0, x_1, x_2, x_3] + \cdots + (x - x_0)\ldots(x - x_{n-1})[x_0, x_1, \ldots, x_n] + \cdots$$

com $x = x_j + kh$ e que é conhecida por *fórmula interpoladora de Newton com diferenças divididas*.

Suponha que a função $f(x)$ e as suas n primeiras derivadas são finitas e contínuas no intervalo $[a, b]$, com $a \leq x_0 < x_1 < \ldots < x_n \leq b$ e $x \in [a, b]$ e que a derivada de ordem $n + 1$, $f^{(n+1)}(x)$, existe. O erro de truncatura, que corresponde a

$$R_n(x) = f(x) - p_n(x),$$

com

$$p_n(x) = f_0+(x-x_0)[x_0,x_1]+(x-x_0)(x-x_1)[x_0,x_1,x_2]+(x-x_0)(x-x_1)(x-x_2)[x_0,x_1,x_2,x_3]$$
$$+\cdots+(x-x_0)\ldots(x-x_{n-1})[x_0,x_1,\ldots,x_n]$$

o polinómio interpolador de grau $\leq n$, cuja derivada de ordem n é dada por

$$p_n^{(n)}(x) = n!\,[x_0,x_1,\ldots,x_{n-1},x_n]\,,$$

anula-se nos $n+1$ pontos $x_0,x_1,\ldots,x_n$. Pelo teorema de Rolle conclui-se que $R_n'(x)$ anula-se em n pontos do intervalo $[a,b]$, $R_n''(x)$ anula-se em $n-1$ pontos de $[a,b]$, ..., e $R^{(n)}(x)$ anula-se num determinado ponto. Seja ξ este ponto. Então, de

$$R^{(n)}(\xi) = f^{(n)}(\xi) - p_n^{(n)}(\xi) = 0 \iff f^{(n)}(\xi) = n!\,[x_0,x_1,\ldots,x_n]$$

tira-se que

$$[x_0,x_1,\ldots,x_n] = \frac{f^{(n)}(\xi)}{n!}$$

que é a fórmula que relaciona a diferença dividida de ordem n com a derivada *da mesma ordem* de $f(x)$ num ponto de $[a,b]$.

Do mesmo modo, tem-se

$$[x_0,x_1,\ldots,x_n,x] = \frac{f^{(n+1)}(\xi)}{(n+1)!}\,,\quad \text{com}\quad \xi\in[a,b]$$

e o erro de truncatura do polinómio interpolador de Newton de grau n, $p_n(x)$ (com diferenças divididas), é dado por

$$R_n(x) = (x-x_0)(x-x_1)\ldots(x-x_{n-1})(x-x_n)\frac{f^{(n+1)}(\xi)}{(n+1)!}\,. \tag{6.13}$$

Uma outra fórmula interpoladora que pode ser usada em interpolação polinomial baseada em $n+1$ pontos não igualmente espaçados, onde não seja necessário calcular diferenças, é a fórmula de Lagrange. Como se verá em 6.3.5. esta fórmula exige uma maior quantidade de cálculos do que a fórmula de Newton.

6.3.3 Interpolação inversa

Além da fórmula de Everett e das aproximações sucessivas (veja-se em 6.2.7), existe um outro processo para resolver o problema da interpolação inversa e que usa diferenças divididas. A fórmula de Newton calcula aproximações polinomiais à função $f(x)$. Invertendo a tabela de valores de f, colocando os valores da função na posição dos argumentos e os valores de x na coluna dos valores da função, e implementando directamente a fórmula interpoladora de Newton com diferenças divididas correspondentes, o resultado traduz-se na aproximação de x por um polinómio. A tabela com as diferenças divididas de x passa a ser a seguinte,

f_0	x_0				
		$[f_0, f_1]$			
f_1	x_1		$[f_0, f_1, f_2]$		
		$[f_1, f_2]$		$[f_0, f_1, f_2, f_3]$	
f_2	x_2		$[f_1, f_2, f_3]$		
		$[f_2, f_3]$		$[f_1, f_2, f_3, f_4]$	$[f_0, f_1, \ldots, f_{n-1}, f_n]$
$\ldots$	$\ldots$				
f_{n-2}	x_{n-2}		$[f_{n-3}, f_{n-2}, f_{n-1}]$		
		$[f_{n-2}, f_{n-1}]$		$[f_{n-3}, f_{n-2}, f_{n-1}, f_n]$	
f_{n-1}	x_{n-1}		$[f_{n-2}, f_{n-1}, f_n]$		
		$[f_{n-1}, f_n]$			
f_n	x_n				

Tabela 6.5: Tabela das diferenças divididas de x

e a fórmula de Newton fica com o seguinte aspecto

$$\begin{aligned} x &= x_0 + (f(x) - f_0)[f_0, f_1] + (f(x) - f_0)(f(x) - f_1)[f_0, f_1, f_2] \\ &+ (f(x) - f_0)(f(x) - f_1)(f(x) - f_2)[f_0, f_1, f_2, f_3] + \cdots \end{aligned}$$

donde se retira, então, directamente, o valor de x para o qual a função toma o valor especificado $f(x)$.

O algoritmo 6.3 resolve quer o problema directo quer o inverso, por interpolação polinomial de Newton, baseado nas diferenças divididas. A selecção é feita através do parâmetro *interp*.

Algoritmo 6.3 :

1. ler n, $grau(\leq n)$ e para $i = 0, 1, \ldots, n$ ler x_i e $f(x_i)$
2. ler *valor*, *interp* ("directa"ou "inversa") e fazer $l = 0$
3. se *interp* = "directa"então
   - 3.1. para $i = 0, \ldots, n$ fazer $d_{i0} = f(x_i)$
   - 3.2. para $j = 1, \ldots, n$ e $i = 0, \ldots, n - j$ calcular $d_{ij} = \frac{d_{i+1\,j-1} - d_{i\,j-1}}{x_{j+i} - x_i}$
   - 3.3. fazer $coef_0 = 1$ e para $i = 1, \ldots, grau$ calcular $coef_i = coef_{i-1}(valor - x_{i-1})$
   - 3.4. calcular $polin = \sum_{j=0}^{grau} coef_j d_{0j}$
   - 3.5. se 'quer interpolar para outro valor de x' então
     - 3.5.1. fazer $l = l + 1$
     - 3.5.2. colocar *polin* em $p_l(valor)$
     - 3.5.3. ir para o passo 3.3 ou 1. (pode precisar de introduzir outros pontos)
   - 3.6. terminar com $p_{l+1}(valor) \leftarrow polin$.
4. senão (interpolação inversa) para $i = 0, \ldots, n$ fazer $d_{i0} = x_i$
5. para $j = 1, \ldots, n$ e $i = 0, \ldots, n - j$ calcular $d_{ij} = \frac{d_{i+1\,j-1} - d_{i\,j-1}}{f(x_{j+i}) - f(x_i)}$

6. `fazer` $coef_0 = 1$ `e para` $i = 1, \ldots, grau$ `calcular` $coef_i = coef_{i-1}(valor - f(x_{i-1}))$

7. `calcular` $argum = \sum_{j=0}^{grau} coef_j d_{0j}$

8. `terminar com` $x \leftarrow argum$.

6.3.4 Implementação. Rotina DIFDIV

A rotina DIFDIV implementa o algoritmo 6.3 e foi usada na resolução dos seguintes exemplos:

Exemplo 6.5 Dada a tabela de doze valores da função $f(x)$

x_i	f_i
0.00	0.0
0.30	0.295520206
0.50	0.479425538
0.70	0.644217687
0.90	0.783326909
1.00	0.841470984
1.20	0.932039086
1.50	0.997494986
1.60	0.999573603
1.75	0.983985946
2.00	0.909297426
2.10	0.863209366

e fazendo $interp =$ directa, a aproximação polinomial de Newton, baseada em diferenças divididas, de grau 3 (usando apenas os quatro pontos mais próximos do ponto interpolador) fornece, para $x = 1.57$, o valor 0.999995.

Se, em vez de quatro pontos, se usarem seis pontos $(1, 1.2, 1.5, 1.6, 1.75, 2)$, a aproximação polinomial de grau 5 dá $p_5(1.57) = 1.000000$.

Finalmente, usando todos os pontos da tabela, o polinómio construído é de grau 11 e $p_{11}(1.57) = 1.000000$.

Para calcular o valor de x para o qual $f(x) = 0.5$, $interp =$ inversa, a rotina DIFDIV fornece $x = 0.523683$ quando os cálculos são baseados nos seis primeiros pontos, a que corresponde uma interpolação polinomial de grau 5. Para $f(x) = 1$, uma aproximação polinomial de grau 5 (seis pontos do intervalo $[1, 2]$) fornece $x = 1.628018$.

Exemplo 6.6 Dada a tabela de valores de $f(x)$,

x_i	0.0	1.0	2.0	3.0
f_i	0.0	0.3	0.7	1.0

obteve-se para $x = 0.1$ o valor $p_3 = 0.019800$, para $x = 0.5$ o valor $p_3 = 0.125000$ e para $x = 1.5$ o valor $p_3 = 0.500000$.

6.3.5 Fórmula interpoladora de Lagrange

Quando se usa a fórmula interpoladora de Newton com diferenças divididas (veja-se em (6.12)) e se substituem as diferenças divididas pelas suas expressões em função dos valores da função tabelados $f_0, f_1, f_2, \ldots, f_n, \ldots$, obtém-se a *fórmula interpoladora de Lagrange*[5],

$$f(x) = L_0(x)f_0 + L_1(x)f_1 + \cdots + L_n(x)f_n + \cdots \tag{6.14}$$

Utilizando apenas $n+1$ pontos $x_0, x_1, \ldots, x_n$ e os correspondentes valores da função, os coeficientes $L_i(x)$ da fórmula (6.14) são polinómios de grau n, em x, dados por

$$L_i(x) = \frac{(x-x_0)\ldots(x-x_{i-1})(x-x_{i+1})\ldots(x-x_n)}{(x_i-x_0)\ldots(x_i-x_{i-1})(x_i-x_{i+1})\ldots(x_i-x_n)}\,, \quad i = 0, 1, \ldots, n$$

e que satisfazem

$$L_i(x_j) = \begin{cases} 1 & \text{se } j = i \\ 0 & \text{se } j \neq i. \end{cases}$$

O polinómio interpolador de Lagrange, também de grau n em x tem a forma

$$p_n(x) = L_0(x)f_0 + L_1(x)f_1 + \cdots + L_n(x)f_n\,,$$

e satisfaz as condições de polinómio de colocação $p_n(x_i) = f_i\,, i = 0, 1, \ldots, n$.

Como o polinómio é o mesmo do que o encontrado pela fórmula de Newton, para o mesmo conjunto de $n+1$ pontos, o erro de truncatura desta aproximação polinomial é dado pelo resto, já anteriormente apresentado na equação (6.13), em 6.3.2.

6.4 Interpolação segmentada

6.4.1 Introdução teórica

Uma 'spline' é uma função segmentada e consiste na junção de várias funções definidas num intervalo, de tal forma que as partes estão ligadas umas às outras de uma maneira contínua e suave. Isto é, existe continuidade na 'spline' nos pontos que unem as partes.

O *conjunto de nós* é um conjunto de pontos $\Gamma_n = \{x_i\}_{i=0}^n$, em que $a = x_0 < x_1 < \ldots < x_{n-1} < x_n = b$, que divide um dado intervalo $[a, b] \subset \mathbf{R}$ em segmentos. Aos

[5] J. L. Lagrange (1736-1813) nasceu em Turin (Itália) e desenvolveu grande parte da sua actividade nas cidades de Paris e Berlim. Lagrange publicou uma vasta obra, no período de 1759 a 1795, compilada e editada por J.-A. Serret e G. Darboux em 1867-1892, 'Oeuvres: Oeuvres de Lagrange' em 14 volumes. As contribuições mais significativas de Lagrange foram nos domínios das Equações Diferença (foi o primeiro a relacionar o cálculo da raiz de uma equação algébrica com equações lineares diferença com coeficientes constantes, em 1759), Séries de Fourier, Equações Diferença Parciais (em que considera duas, três e quatro variáveis independentes e coeficientes constantes num trabalho de 1775), Diferenças Finitas e Interpolação (a fórmula, hoje, conhecida por fórmula de Lagrange foi descoberta em 1794/5) e Interpolação Trigonométrica para comportamentos periódicos. Foi também um dos primeiros criadores do Cálculo das Variações. Lagrange era muito respeitado, tendo, em 1793, presidido à Comissão de normalização de pesos e medidas (Goldstine (1977), Kahaner, Moler e Nash (1989) e Hämmerlin e Hoffmann (1991)).

pontos $x_1, x_2, \ldots, x_{n-1}$ chamam-se *nós interiores* e aos x_0 e x_n chamam-se *nós exteriores ou fronteiras*.

Seja l um inteiro não negativo. Uma função $s_l : [a,b] \to \mathbf{R}$ chama-se *'spline' de grau* l se possuir as seguintes propriedades:

a) $s_l \in C_{l-1}[a,b]$, isto é, s_l é uma função continuamente diferenciável até à ordem $l-1$,

b) $s_l \in P_l$ (é um polinómio de grau l) para $x \in [x_i, x_{i+1}],\ 0 \leq i \leq n-1$.

6.4.2 'Splines' linear e cúbica, natural e completa

Uma 'spline' linear, ou de primeira ordem, é definida (para um conjunto de pontos ordenados) por um conjunto de polinómios de grau um ligados entre si nos nós interiores. Considerem-se os $n+1$ pontos ordenados de tal forma que

$$x_0 < x_1 < x_2 < \ldots < x_{n-1} < x_n$$

e os correspondentes valores da função $f(x_0), f(x_1), \ldots, f(x_n)$ e, designe-se o intervalo $[x_{i-1}, x_i]$ por *segmento* i $(i = 1, 2, \ldots, n)$. O polinómio de grau um do segmento i que passa pelos extremos do intervalo tem a seguinte forma:

$$s_1^i(x) = f(x_{i-1}) + \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}}(x - x_{i-1}) \quad \text{para} \quad i = 1, 2, \ldots, n \tag{6.15}$$

A 'spline' linear resultante é uma função contínua no intervalo $[x_0, x_n]$, no entanto, a primeira derivada é descontínua nos nós interiores (pontos de união dos polinómios).

O teorema seguinte estabelece um limite superior para o erro de truncatura da interpolação 'spline' linear.

Teorema 6.3 : Seja $f(x)$ uma função contínua com derivadas contínuas até à segunda ordem, no intervalo $[a,b]$, com $a = x_0 < x_1 < \ldots < x_{n-1} < x_n = b$. Seja $s_1(x)$ uma função 'spline', composta pelos polinómios $s_1^i(x)$, $i = 1, \ldots, n$, linear para aproximar $f(x)$ no intervalo $[a,b]$. Se

$$máx_{\xi \in [a,b]} |f''(\xi)| \leq M_2$$

então

$$|f(x) - s_1(x)| \leq \frac{1}{8} h^2 M_2, \tag{6.16}$$

para $x \in [a,b]$, em que

$$h = máx_{0 \leq i \leq n-1}(x_{i+1} - x_i).$$

Para se assegurar que as derivadas de uma função 'spline' de ordem igual ou inferior a m sejam contínuas, devem-se construir, em cada segmento, polinómios de grau igual ou superior a $m+1$. A 'spline' resultante diz-se de ordem $m+1$. Assim, por exemplo, a 'spline'

cúbica é uma função contínua com primeira e segunda derivadas contínuas. Veremos, agora como construir uma 'spline' cúbica.

Em cada segmento i, pretende-se construir um polinómio de grau três, s_3^i, que passa pelos extremos do segmento, $[x_{i-1}, x_i]$. A segunda derivada deste polinómio é uma forma linear, que pode ser construída a partir dos polinómios de Lagrange, a passar pelos dois pontos x_{i-1} e x_i,

$$s_3^{i\,\prime\prime} = \frac{x - x_i}{x_{i-1} - x_i} M(x_{i-1}) + \frac{x - x_{i-1}}{x_i - x_{i-1}} M(x_i).$$

Os $M(x_{i-1})$ e $M(x_i)$ são os valores da segunda derivada da função naqueles pontos, embora sejam desconhecidos. Integrando $s_3^{i\,\prime\prime}$, duas vezes, em ordem a x, obtém-se uma forma polinomial cúbica e que é precisamente a expressão da 'spline' cúbica do segmento i, s_3^i:

$$s_3^i = \frac{M(x_{i-1})}{6(x_i - x_{i-1})}(x_i - x)^3 + \frac{M(x_i)}{6(x_i - x_{i-1})}(x - x_{i-1})^3 + [\frac{f(x_{i-1})}{(x_i - x_{i-1})} - \frac{M(x_{i-1})(x_i - x_{i-1})}{6}]$$

$$(x_i - x) + [\frac{f(x_i)}{(x_i - x_{i-1})} - \frac{M(x_i)(x_i - x_{i-1})}{6}](x - x_{i-1}) \text{ para } i = 1, 2, \ldots, n. \qquad (6.17)$$

Esta expressão do polinómio cúbico do segmento i depende de duas incógnitas: $M(x_{i-1})$ e $M(x_i)$. Para $i = 1$, as incógnitas são $M(x_0)$ e $M(x_1)$; para $i = 2$, elas são $M(x_1)$ e $M(x_2)$; para $i = 3$, elas são $M(x_2)$ e $M(x_3)$, ..., e para $i = n$ as incógnitas são $M(x_{n-1})$ e $M(x_n)$. Assim, considerando o conjunto dos n polinómios de todos os segmentos do intervalo $[a, b]$, tem-se nesta fase, como incógnitas as $n + 1$ quantidades:

$$M(x_0), M(x_1), M(x_2), \ldots, M(x_{n-1}), M(x_n)\,.$$

Para as calcular, deve-se impôr um conjunto de $n + 1$ condições nas 'splines' dos segmentos. A continuidade nas primeiras derivadas nos $n - 1$ nós interiores, $x_1, x_2, \ldots, x_{n-1}$, define as seguintes $n - 1$ condições:

$$s_3^{i\,\prime}(x_i) = s_3^{i+1\,\prime}(x_i) \quad \text{para} \quad i = 1, 2, \ldots, n-1.$$

Vamos deduzir estas condições.

Derivando, uma vez, em ordem a x, a expressão da 'spline' do segmento i (equação (6.17)) e substituindo x por x_i obtém-se

$$s_3^{i\,\prime}(x_i) = \frac{M(x_i)(x_i - x_{i-1})}{3} + \frac{M(x_{i-1})(x_i - x_{i-1})}{6} + \frac{f(x_i)}{(x_i - x_{i-1})} - \frac{f(x_{i-1})}{(x_i - x_{i-1})}\,.$$

Do mesmo modo, derivando a expressão da 'spline' do segmento $i + 1$ (equação (6.17) substituindo i por $i + 1$), em ordem a x, e substituindo posteriormente x por x_i obtém-se

$$s_3^{i+1\,\prime}(x_i) = -\frac{M(x_i)(x_{i+1} - x_i)}{3} - \frac{M(x_{i+1})(x_{i+1} - x_i)}{6} + \frac{f(x_{i+1})}{(x_{i+1} - x_i)} - \frac{f(x_i)}{(x_{i+1} - x_i)}\,.$$

Igualando estas duas expressões fica-se com a equação

$$(x_i - x_{i-1})M(x_{i-1}) + 2(x_{i+1} - x_{i-1})M(x_i) + (x_{i+1} - x_i)M(x_{i+1}) = \frac{6}{(x_{i+1} - x_i)}$$

$$(f(x_{i+1}) - f(x_i)) - \frac{6}{(x_i - x_{i-1})}(f(x_i) - f(x_{i-1})) \quad \text{para o nó } i, \tag{6.18}$$

cujas incógnitas são apenas $M(x_{i-1})$, $M(x_i)$ e $M(x_{i+1})$.

Fazendo $i = 1, \ldots, n-1$ (que correspondem aos nós interiores) em (6.18), obtém-se um sistema com $n-1$ equações nas $n+1$ incógnitas, $M(x_0), M(x_1), \ldots, M(x_{n-1})$ e $M(x_n)$. Este sistema poderá ser possível e determinado se lhe forem adicionadas mais duas equações. Estas, vão depender do tipo de 'spline' que se pretende construir.

Assim, se se optar por uma 'spline' *natural* adicionam-se, ao sistema resultante de (6.18), as equações

$$M(x_0) = 0 \ \text{ e } \ M(x_n) = 0\,, \tag{6.19}$$

que correspondem a exigir que as 'splines' dos primeiro e último segmentos tenham curvatura nula nos nós exteriores, respectivamente, em x_0 e x_n. Isto é, $s_3^{1\,''}(x_0) = 0$ e $s_3^{n\,''}(x_n) = 0$, que correspondem a $M(x_0) = M(x_n) = 0$. Na prática, substituem-se estes valores no sistema que continua a ter $n-1$ equações, no entanto, passam a existir apenas $n-1$ incógnitas.

Para construir uma 'spline' *completa*, deve-se exigir que a primeira derivada da 'spline' do primeiro segmento, calculada em x_0, seja igual a $f'(x_0)$, bem como, a primeira derivada da 'spline' do último segmento, calculada em x_n, seja igual a $f'(x_n)$:

$$s_3^{1\,'}(x_0) = f'(x_0) \text{ e } s_3^{n\,'}(x_n) = f'(x_n)\,.$$

Usando uma dedução idêntica à usada para se chegar a (6.18), obtêm-se as duas equações

$$2(x_1 - x_0)M(x_0) + (x_1 - x_0)M(x_1) = \frac{6}{(x_1 - x_0)}(f(x_1) - f(x_0)) - 6f'(x_0) \tag{6.20}$$

$$2(x_n - x_{n-1})M(x_n) + (x_n - x_{n-1})M(x_{n-1}) = 6f'(x_n) - \frac{6}{(x_n - x_{n-1})}(f(x_n) - f(x_{n-1}))\,. \tag{6.21}$$

O sistema de equações lineares, que no caso da 'spline' natural é de ordem $n-1$ e no caso da 'spline' completa é de ordem $n+1$, é sempre tridiagonal. A sua solução fornece os $M(x_1), \ldots, M(x_{n-1})$, no caso da primeira, ou os $M(x_0), M(x_1), \ldots, M(x_{n-1}), M(x_n)$, no caso da segunda. A partir destes valores, podem-se construir as formas polinomiais cúbicas a partir da expressão (6.17) e que compõem a 'spline' cúbica.

O algoritmo 6.4 constrói uma 'spline' cúbica natural ou completa consoante o valor atribuído ao parâmetro *tipo* e calcula o seu valor quando $x = valor$.

Quando não for conhecida a derivada de $f(x)$, as quantidades $f'(x_0)$ e $f'(x_n)$, respectivamente em (6.20) e (6.21), podem ser aproximadas por diferenças divididas de primeira ordem, tomando um ponto x_a muito próximo de x_0, mas diferente daquele que se considera como x_1, e um outro x_b muito próximo de x_n e diferente daquele que se toma como x_{n-1}. O passo 3.2 do algoritmo 6.4 considera esta possibilidade.

Algoritmo 6.4 :

1. ler n, *tipo* ("natural" ou "completa"), *valor*, fazer $l = 0$ e para $i = 0, \ldots, n$ ler x_i e $f(x_i)$
2. se *tipo* = "natural" então fazer *ordem* $= n - 1$
   - 2.1. para $j = 1, \ldots, n-1$ calcular $d_j = 2(x_{j+1} - x_{j-1})$ e $b_j = \frac{6}{x_{j+1}-x_j}(f(x_{j+1}) - f(x_j)) + \frac{6}{x_j - x_{j-1}}(f(x_{j-1}) - f(x_j))$
   - 2.2. para $j = 1, \ldots, n-2$ calcular $i_{j+1} = x_{j+1} - x_j$ e $s_j = x_{j+1} - x_j$
3. senão ('spline' completa) fazer *ordem* $= n + 1$
   - 3.1. para $j = 1, \ldots, n-1$ calcular $d_{j+1} = 2(x_{j+1} - x_{j-1})$, $b_{j+1} = \frac{6}{x_{j+1}-x_j}(f(x_{j+1}) - -f(x_j)) - \frac{6}{x_j - x_{j-1}}(f(x_j) - f(x_{j-1}))$, $i_{j+1} = x_j - x_{j-1}$ e $s_{j+1} = x_{j+1} - x_j$
   - 3.2. ler $f'0$ e $f'n$ ou ler (x_a, f_a), (x_b, f_b) e calcular $f'0 = \frac{f_a - f(x_0)}{x_a - x_0}$ e $f'n = \frac{f(x_n) - f_b}{x_n - x_b}$
   - 3.3. calcular $s_1 = (x_1 - x_0)$, $d_1 = 2s_1$, $i_{n+1} = (x_n - x_{n-1})$, $d_{n+1} = 2i_{n+1}$, $b_1 = \frac{6}{s_1}(f(x_1) - f(x_0)) - 6f'0$ e $b_{n+1} = 6f'n - \frac{6}{i_{n+1}}(f(x_n) - f(x_{n-1}))$
4. resolver o sistema tridiagonal de *ordem* linhas (usar o algoritmo 3.4) e
   - 4.1. se *tipo* = "natural" então fazer $M_0 = 0$, $M_n = 0$ e colocar o vector solução de $n-1$ elementos em $M_1, M_2, \ldots, M_{n-1}$
   - 4.2. senão, colocar o vector solução de $n+1$ elementos em $M_0, M_1, \ldots, M_{n-1}, M_n$
5. para $j = 0, \ldots, n$ se $valor < x_j$ então fazer $k = j$ e ir para o passo 6
6. calcular o valor da 'spline' do segmento que contém *valor*:
   calcular $y = x_k - x_{k-1}$ e $spline = \frac{M_{k-1}}{6\,y}(x_k - valor)^3 + \frac{M_k}{6\,y}(valor - x_{k-1})^3 + (\frac{f(x_{k-1})}{y} - \frac{y\,M_{k-1}}{6})(x_k - valor) + (\frac{f(x_k)}{y} - \frac{y\,M_k}{6})(valor - x_{k-1})$
7. se 'quer interpolar para outro valor de x' então
   - 7.1. fazer $l = l + 1$
   - 7.2. colocar *spline* em $s_l(valor)$ e ler novo *valor*
   - 7.3. ir para o passo 5
8. terminar com $s_{l+1}(valor) \leftarrow spline$.

Limites superiores para os erros de truncatura da interpolação 'spline' cúbica e da sua primeira derivada, são dados pelo teorema:

Teorema 6.4 : Seja $f(x)$ uma função contínua com derivadas contínuas até à quarta ordem, no intervalo $[a,b]$, com $a = x_0 < x_1 < \ldots < x_{n-1} < x_n = b$. Seja $s_3(x)$ uma função 'spline', composta pelos polinómios $s_3^i(x)$, $i = 1, \ldots, n$, cúbica completa para aproximar $f(x)$ no intervalo $[a,b]$. Se

$$máx_{\xi \in [a,b]} |f^{(iv)}(\xi)| \leq M_4$$

então

$$|f(x) - s_3(x)| \leq \frac{5}{384} h^4 M_4, \tag{6.22}$$

e

$$|f'(x) - s_3'(x)| \leq \frac{1}{24} h^3 M_4,$$

para $x \in [a,b]$, em que

$$h = máx_{0 \leq i \leq n-1} (x_{i+1} - x_i).$$

6.4.3 Implementação. Rotina SPLINE

A rotina SPLINE implementa o algoritmo 6.4. Os resultados obtidos para os seguintes exemplos são:

Exemplo 6.7 Dada a função $f(x)$ definida para este conjunto de 11 pontos

x_i	f_i
5.0	0.0639269
5.1	0.0800431
5.2	0.0987853
5.3	0.1202508
5.4	0.1444762
5.5	0.1714318
5.6	0.2010186
5.7	0.2330688
5.8	0.2673489
5.9	0.3035673
6.0	0.3413825

e usando uma 'spline' natural obtém-se para $x = 5.45$ o valor 0.157618, que corresponde a uma interpolação cúbica. Como foram definidos 10 segmentos, o polinómio s_3^5 usado na interpolação é o do intervalo $[5.4, 5.5]$.

Usando, agora, apenas os 6 pontos seguintes:

x_i	f_i
5.0	0.0639269
5.2	0.0987853
5.4	0.1444762
5.6	0.2010186
5.8	0.2673489
6.0	0.3413825

definem-se 5 segmentos e o ponto interpolador $x = 5.45$ encontra-se no terceiro segmento. O valor obtido $s_3^3(5.45)$ é 0.157659.

Com esta tabela e aproximando

$$f'(x_0) \text{ por } \frac{f(5.1) - f(5.0)}{5.1 - 5.0}$$

e

$$f'(x_n) \text{ por } \frac{f(6.0) - f(5.9)}{6.0 - 5.9}$$

a rotina SPLINE fornece, através de uma 'spline' completa, para $x = 5.45$, o valor 0.157655.

Exemplo 6.8 Considerando a tabela de doze pontos do exemplo 6.5, a 'spline' cúbica natural calcula, para $x = 1.57$, o valor 1.000009, quase o mesmo que foi dado por um polinómio único de grau 11.

Considerando apenas dez pontos da tabela, excluindo os pontos $(0.3, 0.295520)$ e $(2.0, 0.909297)$, a 'spline' cúbica completa dá o valor 0.999973 para $x = 1.57$. As derivadas $f'(x_0)$ e $f'(x_n)$ foram aproximadas por:

$$f'(x_0) \approx \frac{f(0.3) - f(0.0)}{0.3}$$

e

$$f'(x_n) \approx \frac{f(2.1) - f(2.0)}{2.1 - 2.0}.$$

6.5 Problemas

1. Dada a função $f(x)$ definida pela seguinte tabela de oito pontos

x_i	0.0	0.2	0.4	0.6	0.8	1.0	1.2	1.4
f_i	1.000	1.221	1.492	1.822	2.226	2.718	3.320	4.056

   a) Use o polinómio interpolador de Gregory-Newton de grau 4 para calcular uma aproximação a $f(0.15)$. Calcule um limite superior do erro de truncatura cometido com esta aproximação.

   b) Use o polinómio interpolador de Gregory-Newton de grau 4 para aproximar $f(1.35)$. Estime o erro de truncatura cometido.

   c) Aproxime $f(1.35)$ usando um polinómio de grau 3, baseado em diferenças ascendentes.

   d) Construa um polinómio de grau 3 baseado nos polinómios de Lagrange, com o objectivo de estimar $f(1.35)$. Compare-o com o obtido em c). Comente os resultados.

2. Considere a função $f(x)$ definida pela seguinte tabela de valores:

x_i	-0.2	0.0	0.2	0.4	0.6	0.8	1.0	1.2
f_i	-0.7328	-0.7071	-0.6528	-0.3981	0.5721	3.1165	8.4372	18.0797

a) Construa a tabela das diferenças centrais de f.

b) Determine o zero da função $f(x)$ que pertence ao intervalo $[-0.2, 1.2]$, usando o processo das aproximações sucessivas baseado na fórmula de Everett. Use todas as diferenças disponíveis a não ser que a partir de uma certa ordem elas tenham um valor médio próximo de zero.

3. Dada a tabela de valores de uma função $f(x)$,

x_i	0.0	0.1	0.3	0.4	0.9	1.0
$f(x_i)$	1	0	-1	0	3	4

a) Construa o polinómio interpolador de Newton, de grau 3, que melhor aproxima a função, para estimar o valor de $f(0.2)$.

b) Acha que o polinómio de grau 3 encontrado na alínea a é uma boa aproximação à função $f(x)$?

Justifique a sua afirmação.

c) Use o mesmo polinómio da alínea a para estimar $f(0.95)$. Como explica o resultado obtido ?

d) Quantos zeros tem a função dada pela tabela no intervalo $[0, 1]$?

e) Estime o melhor possível $f(0.2)$ usando um polinómio interpolador de Newton de grau 2.

4. Dada a tabela de valores de uma função $f(x)$

x_i	0.0	0.1	0.2	0.3	0.4	0.5	0.8	1.0
$f(x_i)$	0	1	1	2	2	3	3	4

pretende-se aproximar $f(0.6)$ usando um polinómio de grau 3.

a) Use a fórmula interpoladora de Newton baseada em diferenças divididas.

b) Estime o erro de truncatura cometido na alínea anterior.

c) Use interpolação segmentada baseada numa 'spline' cúbica natural.

5. A função $f(x)$ está definida para o seguinte conjunto de seis pontos

x_i	40	42	45	48	49	50
f_i	1.60206	1.62325	1.65321	1.68124	1.69020	1.69857

a) Construa a tabela das diferenças divididas. Use um polinómio de grau três para aproximar $f(47)$.

b) Calcule um limite superior do erro de truncatura cometido na aproximação de a).

c) Usando uma função 'spline' cúbica natural para aproximar f, estime $f(47)$. Compare com o valor obtido em a).

6. Dada a tabela de valores de uma função $f(x)$

x_i	0.0	0.1	0.3	0.4	0.9	1.0
$f(x_i)$	0	1	2	3	2	3

pretende-se aproximar $f(0.6)$ usando um polinómio de grau 3.

Use interpolação segmentada baseada numa 'spline' cúbica completa.

7. Considere a tabela de sete pontos de $f(x)$

x_i	-1.00	-0.95	-0.85	-0.80	0.20	0.50	0.90
f_i	-1.00	-0.05	0.90	1.00	0.90	0.50	-0.30

a) Usando todos os pontos construa uma função 'spline' cúbica natural para aproximar $f(x)$. Estime $f(0.60)$.

b) Usando apenas cinco pontos, retirando o segundo e o penúltimo da tabela, construa uma função 'spline' cúbica completa para aproximar $f(x)$. Para aproximar $f'(x_0)$ use os dois primeiros pontos da tabela e para aproximar $f'(x_4)$ use os dois últimos pontos da tabela. Finalmente estime $f(0.60)$. Compare este resultado com o obtido em a).

c) Calcule uma estimativa do erro de truncatura cometido em b).

d) Usando os mesmos cinco pontos de b) construa uma função 'spline' linear. Estime $f(0.60)$. Comente os resultados. Que tipo de comportamente parece ter $f(x)$ no intervalo $[0.2, 0.9]$?

Capítulo 7

Aproximação dos mínimos quadrados

7.1 Introdução

7.1.1 Forma geral do problema

Afirmou-se, no Capítulo 6, que na formulação de um problema de aproximação, duas das componentes mais importantes, a ter em conta, eram o tipo e a quantidade de dados existentes.

Quando os dados são obtidos por medições ou leituras directas de uma experiência, vêm afectados de erros (também chamados ruídos). Nesta situação não faz muito sentido exigir que a função aproximação passe pelos pontos dados. É preferível construir uma outra função que reflicta, na generalidade, o comportamento dos dados.

Assim, dada uma função definida num intervalo $[a, b] \in \mathbf{R}$ por

a) uma tabela matemática de valores $(x_i, f_i), i = 1, \ldots, m$, com $a \leq x_1 < x_2 < \ldots < x_{m-1} < x_m \leq b$, ou

b) uma relação funcional $f(x)$,

pretende-se calcular uma função aproximação simples, $M(x)$, e que pode ser o caso particular de um polinómio de grau menor ou igual a n, $p_n(x)$, que aproxima o melhor possível a função dada.

A proximidade de f em relação a $M(x)$, é definida pelo resíduo, $r(x) = f - M(x)$, e a *medida* que vamos considerar neste Capítulo baseia-se no produto escalar.

Definição 7.1 : O produto escalar de duas funções $f(x)$ e $g(x)$ é definido por

$$\langle f(x), g(x) \rangle = \int_a^b w(x) f(x) g(x) dx$$

se as funções f e g forem contínuas no intervalo $[a,b]$, ou

$$\langle f(x), g(x)\rangle = \sum_{i=1}^{m} w(x_i) f(x_i) g(x_i)$$

se as funções f e g forem definidas para um conjunto discreto de pontos, $x_1, x_2, \ldots, x_m$ do intervalo $[a,b]$. A função $w(x)$, conhecida por função peso, é sempre positiva.

Se o objectivo é determinar uma forma simples, em particular a polinomial, que aproxima a função dada o melhor possível, então, pretende-se tornar o resíduo, $r(x)$, o mais pequeno possível. Tomando o produto escalar como *medida*, pretende-se tornar mínimo $\langle r(x), r(x)\rangle$, ou seja

$$\text{minimizar } \langle f - M(x), f - M(x)\rangle.$$

Uma função $M(x)$, calculada por este processo, chama-se *aproximação dos mínimos quadrados*[1].

De acordo com a definição apresentada de produto escalar, o problema consiste em calcular a função $M(x)$ por forma a

$$\text{minimizar } \int_a^b w(x)(f(x) - M(x))^2 dx \tag{7.1}$$

se a função $f(x)$ for contínua no intervalo $[a,b]$, ou

$$\text{minimizar } \sum_{i=1}^{m} w(x_i)(f_i - M(x_i))^2 \tag{7.2}$$

caso a função esteja definida apenas para um conjunto discreto de pontos: (x_i, f_i), $i = 1, \ldots, m$.

[1] O problema foi originalmente equacionado na forma

$$r_i = b_i - \sum_{j=1}^{n} a_{ij} x_j \, , i = 1, \ldots, m$$

em que os r_i são os erros na resolução de um sistema de m equações lineares não homogéneo em n incógnitas, $Ax = b$, com $m > n$. Pretende-se, assim, calcular o conjunto $x = (x_1, x_2, \ldots, x_n)$ que faz, de alguma maneira, com que os r_i sejam tão pequenos quanto possível. Vários matemáticos e astrónomos tentaram resolver este problema: R. Boscovich (1711-1787), J.T. Mayer em 1748 e mais tarde em 1760, P. S. Laplace em 1799 com a 'Mécanique Céleste' publicada em 1805, Legendre em 1805 foi o primeiro a propor que a soma dos quadrados dos erros, r_i, se tornasse mínima e Gauss, em 1809, relacionou a teoria das probabilidades com o método dos mínimos quadrados. Os estudos de Laplace sobre a teoria das probabilidades (1809) ajudaram Gauss no desenvolvimento de uma abordagem correcta ao problema dos mínimos quadrados. Todo o trabalho de Gauss desenvolvido neste domínio foi mais tarde (1887) compilado e editado em alemão: 'Abhandlungen zur methode der kleinsten quadrate'(Berlim). Existem traduções em francês e inglês (Goldstine (1977)).

7.1.2 Características do problema

Vão surgir, neste Capítulo, dois tipos de espaços de funções. O *espaço linear de funções*, que considera a representação de diferentes modelos $M(x)$ como *combinações lineares* de funções de um conjunto especificado M e que são apresentados em 7.2. com o título 'Problema de mínimos quadrados linear'. O *espaço não linear* é apresentado em 7.3. e considera a construção de modelos que se apresentam como funções não lineares de um conjunto de parâmetros a determinar. Do primeiro espaço, são exemplos

- um modelo polinomial cúbico, $M(x) = p_3(x) = c_0 + c_1x + c_2x^2 + c_3x^3$, em que o conjunto M das funções de base é definido por $M = \{1, x, x^2, x^3\}$, e
- um modelo que envolve funções trigonométricas, $M(x) = c_1 + c_2x + c_3sen(2x) + c_4cos(x)$, em que $M = \{1, x, sen(2x), cos(x)\}$,

  e do segundo, os exemplos,

- $M(x) = c_1(1 + e^{-c_2x})$ e
- $M(x) = \frac{1+c_1x}{c_1c_3+c_2}$.

7.1.3 Aspectos estatísticos do método dos mínimos quadrados

Uma das maiores motivações para a utilização do método dos mínimos quadrados deve-se à redução da influência dos erros aleatórios presentes nas medições ou observações de resultados de experiências. Suponha que os valores de uma função $f(x)$ foram medidos em instantes (ou pontos) $x_1, x_2, \ldots, x_m$. Sejam $f_1, f_2, \ldots, f_m$ essas medições. Os 'verdadeiros' valores da função, que se supõe, serem os *valores esperados* das medições, são os $M(x_1), M(x_2), \ldots, M(x_m)$. Suponha ainda que os erros das medições nos diversos instantes são variáveis estocasticamente independentes. Se o símbolo E designa o *valor esperado*, então tem-se

$$E[f_i] = M(x_i)\,,$$

$$E[(f_i - M(x_i))(f_j - M(x_j))] = \begin{cases} 0 & \text{se } i \neq j \\ \sigma^2 & \text{se } i = j \end{cases}$$

em que σ^2 é a variância dos erros das medições. O problema consiste em usar as medições obtidas para estimar os parâmetros que caracterizam o modelo esperado

$$M(x) = \sum_{i=1}^{n} c_i\phi_i(x)$$

em que as funções $\phi_i(x)$ são conhecidas e $m > n$. A aplicação das técnicas matemáticas e estatísticas para a análise de dados experimentais e o ajuste de modelos matemáticos a estes dados, através da estimação dos parâmetros do modelo, chama-se *análise de regressão*. Se a variável aleatória f segue uma distribuição de probabilidade normal, a técnica da máxima

verosimilhança[2] usada para a estimação dos parâmetros do modelo, é equivalente à técnica dos mínimos quadrados.

7.1.4 Índice de algoritmos

São quatro os algoritmos apresentados neste Capítulo.

Algoritmo 7.1 Mínimos quadrados linear na definição de um modelo polinomial baseado em polinómios ortogonais

Algoritmo 7.2 Mínimos quadrados linear na definição de um modelo não polinomial

Algoritmo 7.3 Mínimos quadrados para a definição de um modelo não linear, usando o método Newton

Algoritmo 7.4 Mínimos quadrados para a definição de um modelo não linear, usando a aproximação Gauss-Newton

O algoritmo 7.1 constrói uma forma polinomial de grau n, baseada em m valores da função e usa os polinómios ortogonais para chegar ao modelo. Se o modelo pretendido não é polinomial, mas linear nos parâmetros a estimar, então o algoritmo 7.2 deve ser seleccionado. Quando os modelos a ajustar são não lineares nos parâmetros, são os algoritmos 7.3 e 7.4 que devem ser usados. O último, para problemas de resíduos nulos ou de pequenos resíduos, em que se suspeita que $r(x) \approx 0$ e o primeiro, sempre que o problema for de grandes resíduos ($r(x) \gg 0$).

7.2 Problema de mínimos quadrados linear

7.2.1 Introdução teórica. Equações normais

No problema linear pretende-se aproximar uma função dada f usando um modelo $M(x)$ com a forma

$$M(x) = c_0\phi_0(x) + c_1\phi_1(x) + \cdots + c_{n-1}\phi_{n-1}(x) + c_n\phi_n(x),$$

que se apresenta como uma combinação linear de funções de base conhecidas do conjunto $M = \{\phi_0(x), \phi_1(x), \ldots, \phi_n(x)\}$. Os $c_0, c_1, \ldots c_n$ são os parâmetros cujos valores têm de ser determinados.

Se interessa calcular o conjunto $\{c_i\}_{i=0}^{n}$ de tal forma que

$$c_0\phi_0(x_i) + c_1\phi_1(x_i) + \cdots + c_n\phi_n(x_i) = f_i\,, \text{ para } i = 1, \ldots, m$$

e se $m = n+1$, o sistema de equações resultante é possível e determinado e tem uma única solução. Nesta situação, a técnica utilizada chama-se interpolação e já se falou dela no

[2] A técnica da máxima verosimilhança é uma técnica estatística para estimar parâmetros e consiste em calcular estimadores, como funções dos elementos da amostra, que maximizam a 'função da verosimilhança'. Esta, não é mais do que a função densidade conjunta dos elementos da amostra, tomada como função dos parâmetros.

Capítulo 6. No entanto, se $m > (n+1)$, o sistema tem mais equações do que incógnitas[3] e é *sobredeterminado*. Neste caso, pretende-se apenas que as *equações sejam verificadas aproximadamente* . Uma das técnicas mais importantes para a resolução de sistemas de equações sobredeterminados é a técnica dos mínimos quadrados[4].

Um dos modelos mais simples nos problemas lineares é o polinómio dos mínimos quadrados, que surge da formulação

$$\text{minimizar } \langle f - p_n(x), f - p_n(x) \rangle$$

em que o polinómio $p_n(x) \in \mathcal{P}$, com $\mathcal{P}$ o conjunto dos polinómios, e é definido da seguinte maneira

$$p_n(x) = c_0 + c_1 x + c_2 x^2 + \cdots + c_{n-1} x^{n-1} + c_n x^n \, . \tag{7.3}$$

Neste exemplo, tem-se o conjunto das funções de base, $P_n \subset M$, definido por

$$P_n = \{1, x, x^2, \ldots, x^{n-1}, x^n\}$$

e a determinação dos coeficientes da combinação (7.3) consegue-se calculando as $n+1$ derivadas parciais de $\langle f - p_n(x), f - p_n(x) \rangle$ em ordem aos c_i e igualando-as a zero.

Considere-se o caso do problema discreto, com $w(x) = 1$, em que

$$\langle f - p_n(x), f - p_n(x) \rangle = S = \sum_{i=1}^{m} (f_i - p_n(x_i))^2.$$

Assim,

$$\begin{aligned}
\frac{\partial S}{\partial c_0} &= \sum_{i=1}^{m} (f_i - c_0 - c_1 x_i - \cdots - c_n x_i^n) = 0 \\
\frac{\partial S}{\partial c_1} &= \sum_{i=1}^{m} (f_i - c_0 - c_1 x_i - \cdots - c_n x_i^n) x_i = 0 \\
\frac{\partial S}{\partial c_2} &= \sum_{i=1}^{m} (f_i - c_0 - c_1 x_i - \cdots - c_n x_i^n) x_i^2 = 0 \\
&\cdots \\
\frac{\partial S}{\partial c_n} &= \sum_{i=1}^{m} (f_i - c_0 - c_1 x_i - \cdots - c_n x_i^n) x_i^n = 0
\end{aligned}$$

[3]Usar mais equações do que incógnitas tem como objectivo reduzir a influência dos erros de medição ou observação.

[4]Uma das ferramentas mais úteis para os analistas numéricos e experimentalistas é a técnica dos mínimos quadrados. Devemos associar a esta técnica os seguintes nomes: Gauss, Legendre e Laplace. Foi Legendre quem primeiro publicou sobre os mínimos quadrados, em 1805, no seu trabalho 'Nouvelles méthodes pour la determination des orbites des comètes' (Paris) baseado na ideia de levar ao mínimo a soma dos quadrados dos erros das observações. No entanto, segundo Laplace, Gauss desenvolveu e deu a conhecer a mesma ideia, uns anos antes de Legendre, embora aparecesse publicada apenas em 1809, em 'Theoria motus corporum coelestium in sectionibus conicis solem ambientium'. O próprio Gauss, na edição inglesa deste trabalho, de 1857, afirma usar o princípio dos mínimos quadrados desde 1795 (Goldstine (1977)).

é um sistema de $n+1$ equações lineares, que pode ser colocado na forma

$$\begin{aligned}
c_0 m + c_1 \sum_{i=1}^{m} x_i + \cdots + c_n \sum_{i=1}^{m} x_i^n &= \sum_{i=1}^{m} f_i \\
c_0 \sum_{i=1}^{m} x_i + c_1 \sum_{i=1}^{m} x_i^2 + \cdots + c_n \sum_{i=1}^{m} x_i^{n+1} &= \sum_{i=1}^{m} f_i x_i \\
c_0 \sum_{i=1}^{m} x_i^2 + c_1 \sum_{i=1}^{m} x_i^3 + \cdots + c_n \sum_{i=1}^{m} x_i^{n+2} &= \sum_{i=1}^{m} f_i x_i^2 \\
&\cdots \\
c_0 \sum_{i=1}^{m} x_i^n + c_1 \sum_{i=1}^{m} x_i^{n+1} + \cdots + c_n \sum_{i=1}^{m} x_i^{2n} &= \sum_{i=1}^{m} f_i x_i^n
\end{aligned}$$

para o cálculo dos $n+1$ coeficientes $c_0, c_1, \ldots, c_n$ e que se chama *sistema das equações normais*.

Este problema, tal como foi equacionado, é mal condicionado. A solução deste sistema é muito sensível a pequenas alterações nos dados ou a pequenos erros induzidos pelos cálculos. A matriz dos coeficientes torna-se cada vez mais perto da singularidade à medida que n aumenta. Esta dificuldade pode ser facilmente ultrapassada reformulando o problema, introduzindo polinómios ortogonais na definição de $p_n(x)$. Este caso vai ser analisado já a seguir.

7.2.2 Modelo polinomial. Polinómios ortogonais

Defina-se a aproximação polinomial dos mínimos quadrados, como uma combinação linear de polinómios ortogonais do conjunto $P_n = \{P_0(x), P_1(x), \ldots, P_n(x)\} \subset M$,

$$p_n(x) = c_0 P_0(x) + c_1 P_1(x) + \cdots + c_{n-1} P_{n-1}(x) + c_n P_n(x). \tag{7.4}$$

O sistema das equações normais resultante é bem condicionado e fácil de resolver, como se verá.

Definição 7.2 : Duas funções $f(x)$ e $g(x)$ dizem-se ortogonais se o seu produto escalar for nulo, isto é, se $\langle f(x), g(x) \rangle = 0$.

Definição 7.3 : A sequência $P_0(x), P_1(x), \ldots, P_n(x)$ forma uma sequência de $n+1$ polinómios ortogonais se os polinómios $P_i(x)$, $i = 0, \ldots, n$ forem ortogonais dois a dois e cada $P_i(x)$ for um polinómio de grau exactamente igual a i.

Os polinómios ortogonais satisfazem as seguintes propriedades:

Propriedade 7.1 : Os polinómios ortogonais $P_n(x)$ são também linearmente independentes. Assim, qualquer polinómio $p_n(x)$, de grau n, pode ser expresso na seguinte forma única

$$p_n(x) = c_0 P_0(x) + c_1 P_1(x) + \cdots + c_n P_n(x),$$

como uma combinação linear de uma sequência de polinómios ortogonais.

$P_n(x)$ não só é ortogonal a cada um dos $P_0, P_1, \ldots, P_{n-1}$ mas também a qualquer combinação linear deles e portanto a qualquer polinómio de grau mais baixo do que n.

Propriedade 7.2 : Os polinómios $P_n(x)$ têm zeros reais e distintos que pertencem a $[a, b]$.

Suponha que $z_1, z_2, \ldots, z_r$ são os zeros, de uma certa multiplicidade impar, de $P_n(x)$ e que pertencem a $[a, b]$. Defina-se $Z(x) = (x - z_1) \ldots (x - z_r)$ e calcule-se

$$\int_a^b w(x) P_n(x) Z(x)\, dx$$

que é diferente de zero uma vez que a função integranda não muda de sinal em $[a, b]$. Como $P_n(x)$ é ortogonal a todos os polinómios de grau mais baixo do que n, então $r \geq n$. No entanto, a soma das multiplicidades não pode ser maior do que n; terá de ser igual a n e assim todas as raízes são simples e todas pertencem a $[a, b]$.

Propriedade 7.3 : Os polinómios ortogonais satisfazem a seguinte relação de recorrência,

$$P_{i+1}(x) = A_i\,(x - B_i)\,P_i(x) - C_i\,P_{i-1}(x)\,, \quad \textit{para} \;\; i = 0, 1, \ldots, n-1 \tag{7.5}$$

sendo $P_0(x) = 1$ ($P_{-1}(x) = 0$, por convenção) e os coeficientes da relação, A_i, B_i e C_i, definidos por:

$$A_i = 1, \quad \textit{para todo o } i\,,$$

$$B_i = \frac{\langle x P_i(x), P_i(x)\rangle}{\langle P_i(x), P_i(x)\rangle}\,, \quad \textit{para todo o } i\,, \tag{7.6}$$

$$C_0 = 0 \quad \text{e} \quad C_i = \frac{\langle P_i(x), P_i(x)\rangle}{\langle P_{i-1}(x), P_{i-1}(x)\rangle} \quad \textit{para} \;\; i > 0. \tag{7.7}$$

Considere-se a nova formulação

$$\text{minimizar } S = \langle f - p_n(x), f - p_n(x)\rangle$$

em que $p_n(x) = c_0 P_0(x) + c_1 P_1(x) + \cdots + c_n P_n(x)$ e f é conhecida para um número discreto de pontos. Derivando parcialmente S em ordem aos coeficientes $c_0, c_1, \ldots, c_n$ e igualando a zero cada uma das derivadas,

$$\frac{\partial S}{\partial c_0} = \sum_{i=1}^{m} (f_i - c_0 P_0(x_i) - c_1 P_1(x_i) - \cdots - c_n P_n(x_i)) P_0(x_i) = 0$$

$$\frac{\partial S}{\partial c_1} = \sum_{i=1}^{m} (f_i - c_0 P_0(x_i) - c_1 P_1(x_i) - \cdots - c_n P_n(x_i)) P_1(x_i) = 0$$

$$\frac{\partial S}{\partial c_2} = \sum_{i=1}^{m}(f_i - c_0P_0(x_i) - c_1P_1(x_i) - \cdots - c_nP_n(x_i))P_2(x_i) = 0$$
$$\cdots$$
$$\frac{\partial S}{\partial c_n} = \sum_{i=1}^{m}(f_i - c_0P_0(x_i) - c_1P_1(x_i) - \cdots - c_nP_n(x_i))P_n(x_i) = 0$$

resulta um sistema de $n+1$ equações lineares, cuja forma matricial é a seguinte

$$\begin{pmatrix} \sum_{i=1}^m P_0(x_i)P_0(x_i) & \sum_{i=1}^m P_0(x_i)P_1(x_i) & \ldots & \sum_{i=1}^m P_0(x_i)P_n(x_i) \\ \sum_{i=1}^m P_1(x_i)P_0(x_i) & \sum_{i=1}^m P_1(x_i)P_1(x_i) & \ldots & \sum_{i=1}^m P_1(x_i)P_n(x_i) \\ \sum_{i=1}^m P_2(x_i)P_0(x_i) & \sum_{i=1}^m P_2(x_i)P_1(x_i) & \ldots & \sum_{i=1}^m P_2(x_i)P_n(x_i) \\ \cdot & \cdot & \cdots & \cdot \\ \sum_{i=1}^m P_n(x_i)P_0(x_i) & \sum_{i=1}^m P_n(x_i)P_1(x_i) & \ldots & \sum_{i=1}^m P_n(x_i)P_n(x_i) \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \cdots \\ c_n \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^m f_iP_0(x_i) \\ \sum_{i=1}^m f_iP_1(x_i) \\ \sum_{i=1}^m f_iP_2(x_i) \\ \cdots \\ \sum_{i=1}^m f_iP_n(x_i) \end{pmatrix}$$

Da definição de polinómios ortogonais, todos os elementos da matriz dos coeficientes do sistema da forma $\sum_{i=1}^m P_j(x_i)P_k(x_i)$, com $j \neq k$, *são nulos*, e o sistema reduz-se à forma diagonal, com solução facilmente obtida a partir de

$$c_0 = \frac{\sum_{i=1}^m f_iP_0(x_i)}{\sum_{i=1}^m P_0(x_i)^2}, \qquad c_1 = \frac{\sum_{i=1}^m f_iP_1(x_i)}{\sum_{i=1}^m P_1(x_i)^2},$$
$$, \ \ldots, \quad c_n = \frac{\sum_{i=1}^m f_iP_n(x_i)}{\sum_{i=1}^m P_n(x_i)^2}. \tag{7.8}$$

Usando a propriedade 7.3 e dada uma tabela de m valores de uma função o algoritmo 7.1 constrói uma sequência de polinómios ortogonais e calcula o valor de $p_n(x)$, de grau n, para um certo valor do argumento, dado pelo parâmetro do algoritmo *valor*.

Algoritmo 7.1 :

1. `ler` n, m, *valor* `e para` $j = 1, \ldots, m$ `ler` x_j `e` f_j
2. `fazer` $aux_0 = m$, $P_{0\,m+1} = 1$, $l = 0$, `para` $j = 1, \ldots, m$ `fazer` $P_{0j} = 1$ `e calcular` $num = \sum_{j=1}^m x_j$
3. `para` $i = 0, \ldots, n$ `fazer`
   - 3.1. `se` $i = 0$ `então`
     - 3.1.1. `calcular` $B_0 = \frac{num}{aux_0}$
     - 3.1.2. `para` $j = 1, \ldots, m$ `calcular` $P_{1j} = x_j - B_0$
     - 3.1.3. `calcular` $b = \sum_{j=1}^m f_j$ `e` $c_0 = \frac{b}{aux_0}$
   - 3.2. `senão`
     - 3.2.1. `calcular` $aux_1 = \sum_{j=1}^m P_{ij}^2$, $b = \sum_{j=1}^m f_jP_{ij}$ `e` $c_i = \frac{b}{aux_1}$
     - 3.2.2. `se` $i = n$ `então ir para o passo 4`

3.2.3. calcular $num = \sum_{j=1}^{m} x_j P_{ij}^2$, $B_i = \frac{num}{aux_1}$ e $C_i = \frac{aux_1}{aux_0}$
3.2.4. para $j = 1, \ldots, m$ calcular $P_{i+1\,j} = (x_j - B_i)P_{ij} - C_i\,P_{i-1\,j}$
3.2.5. fazer $aux_0 = aux_1$

4. para $i = 0, \ldots, n$ imprimir c_i
5. fazer $polin = c_0$
6. para $i = 0, \ldots, n-1$ fazer
   6.1. se $i = 0$ então calcular $P_{1\,m+1} = valor - B_0$
   6.2. senão calcular $P_{i+1\,m+1} = (valor - B_i)\,P_{i\,m+1} - C_i\,P_{i-1\,m+1}$
   6.3. calcular $polin = polin + c_{i+1}P_{i+1\,m+1}$
7. se 'quer interpolar para outro valor de x' então
   7.1. fazer $l = l + 1$
   7.2. colocar $polin$ em $p_{n\,l}(valor)$ e ler novo $valor$
   7.3. ir para o passo 5
8. terminar com $p_{n\,l+1}(valor) \leftarrow polin$.

7.2.3 Implementação. Rotina MQPOLI

A rotina MQPOLI implementa o algoritmo 7.1. Considere os dois exemplos seguintes,

Exemplo 7.1 Para a função dada pela tabela

x_i	0.0	0.2	0.4	0.6	0.8	1.0	1.2	1.4
f_i	1.000	1.221	1.492	1.822	2.226	2.718	3.320	4.056

com oito pontos, a rotina MQPOLI calcula os coeficientes c_0, c_1, c_2 e c_3 do polinómio

$$p_3(x) = c_0P_0(x) + c_1P_1(x) + c_2P_2(x) + c_3P_3(x)$$

e gera os valores dos respectivos polinómios ortogonais, no ponto $x = 0.15$. Assim, obtém-se

$$c_0 = 2.231875,\ c_1 = 2.141012,\ c_2 = 1.051042 \text{ e } c_3 = 0.348169,$$

$p_3(x) = 2.231875 + 2.141012(x - 0.7) + 1.051041(x^2 - 1.4x + 0.28) + 0.348169(x^3 - 2.1x^2 + 1.1x - 0.084)$
e $p_3(0.15) = 1.164466$.

Para o mesmo polinómio, $p_3(1.35) = 3.858760$.

Para calcular o polinómio de grau 4 de mínimos quadrados, que melhor se ajusta aos valores da tabela, obtém-se

$$c_0 = 2.231875,\ c_1 = 2.141012,\ c_2 = 1.051042,\ c_3 = 0.348169 \text{ e } c_4 = 0.095289.$$

Os polinómios ortogonais, P_0, P_1, P_2 e P_3 gerados pela fórmula (7.5) são iguais aos anteriores e $P_4 = x^4 - 2.8x^3 + 2.428571x^2 - 0.6559994x + 0.01919988$.

Daqui se tira que $p_4(0.15) = 1.161273$ e $p_4(1.35) = 3.858010$.

Exemplo 7.2 Para a função dada pela tabela

x_i	-1.00	-0.95	-0.85	-0.80	0.20	0.50	0.90
f_i	-1.00	-0.05	0.90	1.00	0.90	0.50	-0.30

com sete pontos, os coeficientes do polinómio $p_3(x)$ definido por

$$p_3(x) = c_0P_0(x) + c_1P_1(x) + c_2P_2(x) + c_3P_3(x)$$

são

$$c_0 = 0.278571,\ c_1 = 0.052627,\ c_2 = -1.854538 \text{ e } c_3 = 3.276383.$$

Os valores calculados para $p_3(0.0)$ e $p_3(0.6)$ são respectivamente 1.813778 e -0.030680.

7.2.4 Problemas lineares dos mínimos quadrados contínuos

Quando, num problema linear de mínimos quadrados, a definição de produto escalar envolve o integral, a minimização de

$$S = \langle f - p_n(x), f - p_n(x)\rangle = \int_a^b (f(x) - p_n(x))^2 dx$$

com $p_n(x) = c_0P_0(x) + c_1P_1(x) + \cdots + c_nP_n(x)$, pressupõe que a função $f(x)$ é contínua no intervalo $[a, b]$. Considerou-se novamente que $w(x) = 1$. A dedução do sistema das equações normais é feita seguindo os mesmos passos do caso anterior. O somatório é substituído pelo integral entre a e b. Em virtude dos polinómios $P_i(x), (i = 0, \ldots, n)$ serem ortogonais dois a dois, o sistema tem novamente a forma diagonal e a solução é,

$$c_0 = \frac{\int_a^b f(x)P_0(x)dx}{\int_a^b P_0(x)^2dx}, \qquad c_1 = \frac{\int_a^b f(x)P_1(x)dx}{\int_a^b P_1(x)^2dx},$$

$$, \ldots, \quad c_n = \frac{\int_a^b f(x)P_n(x)dx}{\int_a^b P_n(x)^2dx}. \tag{7.9}$$

Neste tipo de problemas contínuos existem casos especiais de polinómios ortogonais que são muito úteis na definição de aproximações polinomiais dos mínimos quadrados. O primeiro caso diz respeito aos *polinómios ortogonais de Legendre*[5]. Eles surgem naturalmente da definição de produto escalar (7.1), quando se tem a função peso $w(x) = 1$ e o intervalo de integração $[-1, 1]$. A relação de recorrência em (7.5), com (7.6) e (7.7) torna-se mais simples

$$P_{i+1}(x) = \frac{2i+1}{i+1}\, x\, P_i(x) - \frac{i}{i+1}P_{i-1}(x)\,, i = 0, 1, \ldots, n-1. \tag{7.10}$$

A tabela 7.1 apresenta na primeira coluna os cinco primeiros polinómios de Legendre.

[5] A. M. Legendre desenvolveu grande parte do seu trabalho em interpolação, na preparação de tabelas trigonométricas e em integração. Em 1785, no trabalho 'Recherches sur l'attraction des sphéroides homogènes' Legendre introduziu os polinómios ortogonais de Legendre. O seu nome está também associado à técnica dos mínimos quadrados (Goldstine (1977)).

1	1	1
x	x	$2x$
$3/2(x^2 - 1/3)$	$2x^2 - 1$	$4x^2 - 2$
$5/2(x^3 - 3/5x)$	$4x^3 - 3x$	$8x^3 - 12x$
$35/8(x^4 - 6/7x^2 + 3/35)$	$8x^4 - 8x^2 - 1$	$16x^4 - 48x^2 + 12$

Tabela 7.1: Polinómios de Legendre, Chebyshev e Hermite

Se o intervalo de interesse continuar a ser $[-1, 1]$, mas a função peso for $\frac{1}{\sqrt{1-x^2}}$, os polinómios ortogonais resultantes são conhecidos por *polinómios ortogonais de Chebyshev*. A função $cos(n\theta)$ é um polinómio em $cos(\theta)$; de facto, se $c = cos(\theta)$ então $cos(2\theta) = 2c^2 - 1$, $cos(3\theta) = 4c^3 - 3c$, ... Escrevendo

$$T_n(x) = cos(n\, cos^{-1}(x))$$

tem-se $T_n(x)$ como um polinómio de grau n:

$$T_0(x) = 1\,,\ \ T_1(x) = x\,,\ \ T_2(x) = 2x^2 - 1\,,\ \ T_3(x) = 4x^3 - 3x\,, \ldots$$

Se o intervalo de integração for outro, mais geral, $[a, b]$, faz-se uma mudança de variáveis

$$t = \frac{(2x - a - b)}{(b - a)}$$

e define-se um novo polinómio $T_n(t) = T_n(a, b; x)$.

Os polinómios de Chebyshev satisfazem propriedades muito importantes,

Propriedade 7.4 :

$$|T_n(x)| \leq 1 \ \ se \ \ |x| \leq 1$$

uma vez que $cos^{-1}(x)$ *é real. Se* $|x| > 1$, $cos^{-1}(x)$ *torna-se complexo, e embora* $T_n(x)$ *continue real, o seu valor é maior do que um.*

Propriedade 7.5 : $T_n(x)$ *atinge os limites* ± 1 *em* $n + 1$ *pontos do intervalo* $[-1, 1]$,

$$T_n(x_j) = (-1)^j \ \ quando \ \ x_j = cos(\frac{j\pi}{n})\,, j = 0, 1, \ldots, n.$$

Propriedade 7.6 :

$$T_n(x_r) = 0 \ \ quando \ \ x_r = cos(\frac{(r + \frac{1}{2})\pi}{n})\,, \ \ r = 1, 2, \ldots, n.$$

Propriedade 7.7 : Estes polinómios são facilmente gerados a partir da relação de recorrência

$$T_{i+1}(x) = 2\, x\, T_i(x) - T_{i-1}(x)\,, \ \ i = 0, 1, \ldots, n - 1$$

retirada de (7.5).

Os cinco primeiros elementos da sequência aparecem na segunda coluna da tabela 7.1.

Propriedade 7.8 :

$$\int_{-1}^{1} \frac{1}{\sqrt{1-x^2}} T_i(x)T_j(x)\,dx = 0 \;\; se \;\; i \neq j.$$

Os *polinómios ortogonais de Hermite*[6] surgem da seguinte formulação

$$\text{minimizar} \int_{-\infty}^{+\infty} e^{-x^2}(f(x) - p_n(x))^2 dx\,.$$

A terceira coluna da tabela 7.1 apresenta os cinco primeiros polinómios de Hermite.

Finalmente, os polinómios ortogonais de Laguerre podem ser gerados a partir da seguinte versão reduzida da relação de recorrência (7.5),

$$P_{i+1}(x) = \frac{1}{i+1}(-x + 2i + \alpha + 1)P_i(x) - \frac{i+\alpha}{i+1}P_{i-1}(x)\,, \;\; i = 0, 1, \ldots, n-1$$

e a formulação do problema tem o seguinte aspecto

$$\text{minimizar} \int_{0}^{+\infty} x^{\alpha} e^{-x}(f(x) - p_n(x))^2 dx\,.$$

7.2.5 Modelo não polinomial

Nesta categoria de aproximação de mínimos quadrados incluem-se várias formas de funções. Por exemplo, funções exponenciais, trigonométricas, do tipo x^{-k}, ... Neste caso a definição do conjunto das funções de base é óbvia. Assim, o formato da função, no espaço das funções lineares, é ainda válido

$$M(x) = c_1\phi_1(x) + c_2\phi_2(x) + \cdots + c_n\phi_n(x) \tag{7.11}$$

com $M = \{\phi_1(x), \phi_2(x), \ldots, \phi_n(x)\}$ e pretende-se calcular o conjunto dos parâmetros $\{c_i\}_{i=1}^n$ que melhor ajusta a função $M(x)$ aos dados do problema, no sentido dos mínimos quadrados, isto é, por forma a minimizar a soma dos quadrados dos resíduos dos f_i em relação aos $M(x_i)$,

$$\text{minimizar} \sum_{i=1}^{m}(f_i - c_1\phi_1(x_i) - c_2\phi_2(x_i) - \cdots - c_n\phi_n(x_i))^2. \tag{7.12}$$

[6] C. Hermite foi um dos matemáticos notáveis do século XIX. Publicou uma vasta obra compilada e editada em Paris no período 1905-1917, por E. Picard, 'Oeuvres: Oeuvres de Charles Hermite'. Destacam-se os seus trabalhos nos domínios da interpolação e integração. Um dos seus trabalhos mais importantes foi publicado em Paris, em 1859, com o título 'Sur l'interpolation'. Em 1875 Hermite estabeleceu algumas das propriedades dos polinómios de Bernoulli. Em 1876 e mais tarde em 1895 publicou alguns resultados relacionados com os números de Bernoulli. O aparecimento dos polinómios ortogonais de Hermite a partir da relação de recorrência data de 1802 (Goldstine (1977)).

Neste caso, supõe-se que as funções $\phi_i(x)$, $i = 1, \ldots, n$, são linearmente independentes, o que é equivalente a supor que a matriz dos coeficientes do sistema das equações normais é não singular. O problema da determinação dos parâmetros tem uma solução única.

Também se considera que os pontos x_i, $i = 1, \ldots, m$, onde são conhecidos os valores da função, f_i, são distintos.

A dedução do sistema linear das equações normais é feita usando os mesmos passos dos dois casos anteriores. Derivando parcialmente a soma, S, em ordem aos c_i, $i = 1, \ldots, n$ e igualando cada uma das derivadas a zero, obtém-se

$$\frac{\partial S}{\partial c_1} = \sum_{i=1}^{m} (f_i - c_1\phi_1(x_i) - c_2\phi_2(x_i) - \cdots - c_n\phi_n(x_i))\phi_1(x_i) = 0$$

$$\frac{\partial S}{\partial c_2} = \sum_{i=1}^{m} (f_i - c_1\phi_1(x_i) - c_2\phi_2(x_i) - \cdots - c_n\phi_n(x_i))\phi_2(x_i) = 0$$

$$\cdots$$

$$\frac{\partial S}{\partial c_n} = \sum_{i=1}^{m} (f_i - c_1\phi_1(x_i) - c_2\phi_2(x_i) - \cdots - c_n\phi_n(x_i))\phi_n(x_i) = 0$$

o sistema das equações normais e lineares, que na forma matricial é

$$\begin{pmatrix} \sum_{i=1}^m \phi_1(x_i)\phi_1(x_i) & \sum_{i=1}^m \phi_1(x_i)\phi_2(x_i) & \ldots & \sum_{i=1}^m \phi_1(x_i)\phi_n(x_i) \\ \sum_{i=1}^m \phi_2(x_i)\phi_1(x_i) & \sum_{i=1}^m \phi_2(x_i)\phi_2(x_i) & \ldots & \sum_{i=1}^m \phi_2(x_i)\phi_n(x_i) \\ . & . & \ldots & . \\ \sum_{i=1}^m \phi_n(x_i)\phi_1(x_i) & \sum_{i=1}^m \phi_n(x_i)\phi_2(x_i) & \ldots & \sum_{i=1}^m \phi_n(x_i)\phi_n(x_i) \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \ldots \\ c_n \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^m f_i\phi_1(x_i) \\ \sum_{i=1}^m f_i\phi_2(x_i) \\ \ldots \\ \sum_{i=1}^m f_i\phi_n(x_i) \end{pmatrix}$$

A solução deste sistema é o vector $(c_1, c_2, \ldots, c_n)^T$ que contém os coeficientes do modelo (7.11).

Determine-se o sistema das equações normais para cada um destes exemplos:

Exemplo 7.3 Considere-se a seguinte função de aproximação

$$M(x) = c_1 + c_2 cos(x) + c_3 sen(x)$$

em que $M = \{1, cos(x), sen(x)\}$. Para calcular os coeficientes de $M(x)$, que melhor ajustam o modelo à função $f(x)$, no sentido dos mínimos quadrados, resolve-se o seguinte sistema das equações normais,

$$\begin{pmatrix} m & \sum_{i=1}^m cos(x_i) & \sum_{i=1}^m sen(x_i) \\ \sum_{i=1}^m cos(x_i) & \sum_{i=1}^m cos^2(x_i) & \sum_{i=1}^m cos(x_i)sen(x_i) \\ \sum_{i=1}^m sen(x_i) & \sum_{i=1}^m sen(x_i)cos(x_i) & \sum_{i=1}^m sen^2(x_i) \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^m f_i \\ \sum_{i=1}^m f_i cos(x_i) \\ \sum_{i=1}^m f_i sen(x_i) \end{pmatrix}.$$

Exemplo 7.4 Considere-se a seguinte formulação:

$$\text{minimizar } \sum_{i=1}^{m}[f_i - (c_1 e^{-x} + c_2 \frac{1}{x})]^2$$

em que $M = \{e^{-x}, \frac{1}{x}\}$. O sistema das equações normais, para o cálculo dos c_1 e c_2 é

$$\begin{pmatrix} \sum_{i=1}^{m} e^{-2x_i} & \sum_{i=1}^{m} e^{-x_i}\frac{1}{x_i} \\ \sum_{i=1}^{m} \frac{1}{x_i}e^{-x_i} & \sum_{i=1}^{m} \frac{1}{x_i^2} \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{m} f_i e^{-x_i} \\ \sum_{i=1}^{m} f_i \frac{1}{x_i} \end{pmatrix}.$$

O algoritmo 7.2 calcula os parâmetros de qualquer modelo $M(x)$, linear e não polinomial, desde que sejam introduzidas as funções de base $\phi_i(x)$ (passo 2) que definem o modelo (7.11).

Algoritmo 7.2 :

1. ler n, m, *valor* e para $j = 1, \ldots, m$ ler x_j e f_j
2. para $i = 1, \ldots, n$ introduzir as funções $\phi_i(x)$
3. para $i = 1, \ldots, n$ e $j = 1, \ldots, m$ calcular $\phi_i(x_j)$
4. definir as equações normais:
   para $i = 1, \ldots, n$
   - 4.1. calcular $b_i = \sum_{j=1}^{m} f_j \phi_i(x_j)$
   - 4.2. para $k = i, \ldots, n$
     - 4.2.1. calcular $a_{ik} = \sum_{j=1}^{m} \phi_i(x_j)\phi_k(x_j)$
     - 4.2.2. se $k \neq i$ fazer $a_{ki} = a_{ik}$
5. resolver o sistema das n equações normais $Ac = b$ (algoritmo 3.3), colocar a solução do sistema em c_i, $i = 1, \ldots, n$ e fazer $l = 0$
6. para $i = 1, \ldots, n$ imprimir c_i
7. para $i = 1, \ldots, n$ calcular $\phi_i(valor)$
8. calcular $modelo = \sum_{i=1}^{n} c_i \, \phi_i(valor)$
9. se 'quer interpolar para outro valor de x' então
   - 9.1. fazer $l = l + 1$
   - 9.2. colocar *modelo* em $M_l(valor)$ e ler novo *valor*
   - 9.3. ir para o passo 7
10. terminar com $M_{l+1}(valor) \leftarrow modelo$.

7.2.6 Implementação. Rotina MQLINE

A implementação do algoritmo 7.2 origina a rotina MQLINE.

Exemplo 7.5 Para a função dada pela tabela do exemplo 7.1

x_i	0.0	0.2	0.4	0.6	0.8	1.0	1.2	1.4
f_i	1.000	1.221	1.492	1.822	2.226	2.718	3.320	4.056
$ln(f_i)$	0.0	0.199670	0.400118	0.599935	0.800206	0.999896	1.199965	1.400197

pretende-se calcular os coeficientes c_1 e c_2 do modelo

$$M(x) = c_1\, e^{c_2 x}$$

por forma a que o ajuste seja o melhor possível. Embora este modelo seja não linear nos parâmetros c_1 e c_2, é possível, através de uma transformação de variáveis, linearizá-lo. Assim,

$$ln(M(x)) = ln(c_1) + c_2 x = C_1 + c_2 x$$

donde se conclui que $\phi_1(x) = 1$ e $\phi_2(x) = x$. Os parâmetros a determinar pelo algoritmo 7.2 são C_1 e c_2. Em virtude da transformação de variáveis, os valores observados que devem ser usados nos cálculos são $ln(f_i)$, $i = 1, \ldots, n$, terceira linha da tabela, em vez dos f_i.

A rotina MQLINE fornece $C_1 = -0.000104$, $c_2 = 1.000147$ e $ln(M(0.15)) = 0.149918$. O modelo $M(x)$ aproxima $f(0.15)$ com o valor 1.161739. Se $x = 1.35$ MQLINE dá 1.350094 e $M(1.35) = 3.857788$.

Exemplo 7.6 Considerando a função definida pela tabela de sete pontos do exemplo 7.2 e o modelo $M(x) = c_1 sen(x) + c_2 cos(x)$, a rotina MQLINE calcula os parâmetros $c_1 = 0.027793$ e $c_2 = 0.512082$. A partir deste modelo estima-se o valor de $f(0.6)$. O valor obtido é $M(0.6) = 0.438333$.

Se em vez deste modelo, se tentar ajustar o modelo $M(x) = c_1 x + c_2 sen(x) + c_3 x^2$, que depende de três parâmetros, a técnica dos mínimos quadrados fornece (algoritmo 7.2) os seguintes valores para $c_1 = 3.768313$, $c_2 = -4.375075$ e $c_3 = 0.062319$. O valor calculado para $M(0.6)$ é -0.186930. Embora, quer este valor quer o anterior se encaixem nos valores da tabela, são muito diferentes. O melhor modelo é aquele cujo valor de $\sum_{i=1}^{m}(f_i - M(x_i))^2$, para os c_i, $i = 1, \ldots, n$ calculados, está mais próximo de zero.

7.2.7 Propriedades da solução

O estudo, que é vulgarmente conhecido por *análise de dados*, tem dois objectivos: o primeiro, serve para determinar o modelo que melhor descreve o comportamento dos dados e, o segundo, diz respeito ao cálculo dos parâmetros que caracterizam o modelo. Neste livro, tratamos essencialmente do segundo aspecto, que nos leva à determinação dos parâmetros e que pressupõe já a existência do modelo. Na prática, esta situação é muito corrente. No entanto, existem casos em que os dados são usados na determinação do modelo.

Quando existe uma utilização dos dados para a definição do próprio modelo, surgem certas questões, às quais convém dar resposta, tais como:

- O modelo descreve adequadamente o comportamento dos dados?

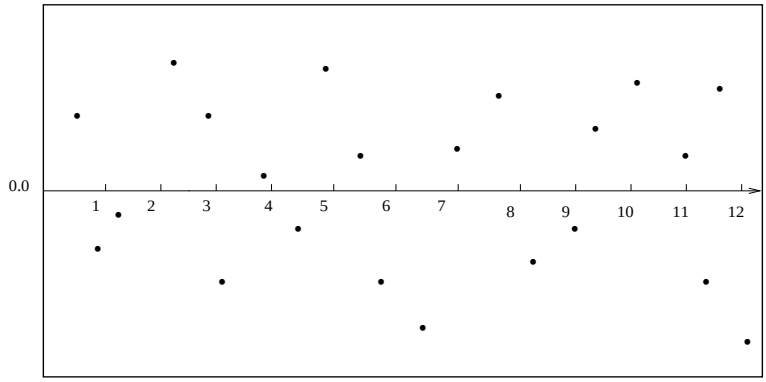


Figura 7.1: Gráfico de resíduos normal

- Existem, no modelo, termos redundantes e que poderiam ser eliminados?
- Os valores calculados para os parâmetros têm a precisão exigida?
- Qual a precisão dos valores previstos pelo modelo?

As respostas a estas questões podem ser obtidas através de *testes estatísticos*. As conclusões a retirar destes testes serão válidas se for possível admitir que os erros existentes nos dados são aleatórios, independentes e normalmente distribuídos.

Um dos processos estatísticos mais simples, para detectar problemas no modelo, consiste na análise do gráfico dos resíduos, $r(x_i) = f_i - M(x_i; c_1^*, c_2^*, \ldots, c_n^*)$, $i = 1, \ldots, m$, calculados em função dos parâmetros obtidos. Uma mancha aleatoriamente distribuída no plano significa que o modelo é adequado. Veja-se o exemplo da figura 7.1.

Os *resíduos escalonados* são vulgarmente utilizados para identificar problemas no modelo e/ou nos próprios valores de f_i. Se s_r for o *desvio padrão dos resíduos*, $r(x_i)$,

$$s_r = \sqrt{\frac{\sum_{i=1}^{m}(f_i - M(x_i; c_1^*, \ldots, c_n^*))^2}{m-n}}$$

define-se resíduo escalonado, $r_{esc}(x_i)$, por

$$r_{esc}(x_i) = \frac{f_i - M(x_i; c_1^*, \ldots, c_n^*)}{s_r},$$

para $i = 1, \ldots, m$.

Aproximadamente 95% dos resíduos escalonados devem pertencer ao intervalo $[-2, 2]$. Se existirem valores fora deste intervalo, os dados que lhes correspondem devem ser analisados com cuidado, uma vez que podem dar indicações sobre a existência de erros nos dados ou no modelo. Gráficos de resíduos escalonados que identificam problemas no modelo e/ou nos dados estão representados na figura 7.2.

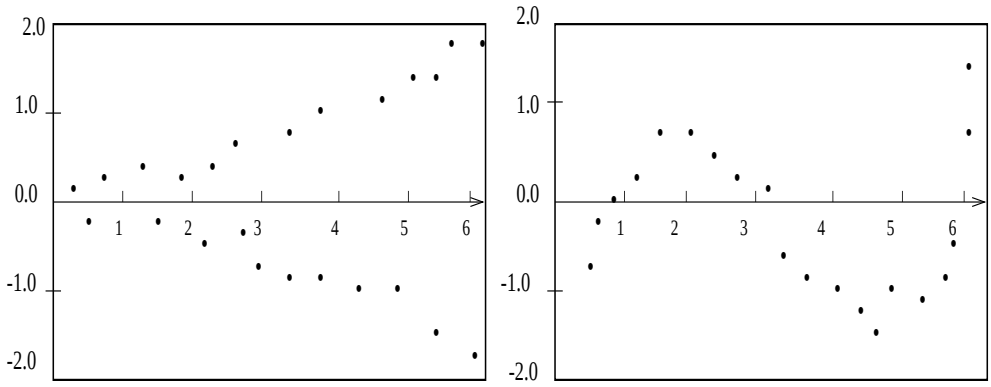


Figura 7.2: Gráficos de resíduos escalonados suspeitos

O *desvio padrão dos dados*, s_f, uma medida da dispersão das observações f_i, dá indicações sobre a distribuição dos f_i. Se as observações forem normalmente distribuídas, 95% das observações estão no intervalo definido por $[\bar{f} - 2\,s_f, \bar{f} + 2\,s_f]$, em que

$$s_f = \sqrt{\frac{\sum_{i=1}^{m}(f_i - \bar{f})^2}{m-1}}$$

e

$$\bar{f} = \frac{\sum_{i=1}^{m} f_i}{m}.$$

A *matriz das variâncias e covariâncias*, dada pela inversa da matriz dos coeficientes do sistema das equações normais, serve para medir a variação e a interdependência entre os parâmetros. O elemento k da diagonal é a variância do parâmetro c_k; a variância é o quadrado do desvio padrão. O elemento (j,k), $j \neq k$, da matriz é a covariância dos parâmetros c_j e c_k. Se forem completamente independentes a sua covariância será nula. Se os dois parâmetros estão relacionados, de alguma maneira, a covariância será diferente de zero. Numa situação ideal os parâmetros deveriam ser independentes e os elementos fora da diagonal principal, da matriz das variâncias e covariâncias, devem ser muito pequenos.

7.2.8 Aproximação de funções periódicas

Em certos domínios da engenharia, ocorrem frequentemente funções periódicas e surge, então, a necessidade de aproximar estas funções a partir de observações.

Considere a função $f(x)$ contínua no intervalo $[-\pi, \pi]$ e que satisfaz a condição de periodicidade:

$$f(x + 2\pi) = f(x), \text{ para } -\infty < x < \infty.$$

Se o período de $f(x)$ é um valor p, diferente de 2π, uma mudança de variável do tipo $t = \frac{2\pi x}{p}$ origina uma função $g(t) = f(\frac{pt}{2\pi})$ que tem período 2π.

As funções de aproximação periódicas são do tipo

$$M_p(x) = \frac{1}{2}c_0 + \sum_{j=1}^{n}[c_j\, cos(jx) + c'_j\, sen(jx)],\ \ -\infty < x < \infty, \tag{7.13}$$

em que os c_j, $j = 0, 1, \ldots, n$ e os c'_j, $j = 1, \ldots, n$ são $2n+1$ parâmetros reais a determinar. O conjunto das funções $M_p(x)$ define o espaço dos polinómios trigonométricos de grau n. O *grau* do polinómio trigonométrico (7.13) é o maior inteiro j tal que, pelo menos, um dos coeficientes, c_j e c'_j, é diferente de zero.

O teorema seguinte estabelece que é possível aproximar, qualquer função periódica e contínua com uma certa precisão, por um polinómio trigonométrico, desde que se escolha n suficientemente grande.

Teorema 7.1 : Para qualquer função $f : \mathbb{R} \to \mathbb{R}$ contínua e periódica, de período 2π, e para qualquer $\varepsilon > 0$, existe um polinómio da forma (7.13) que satisfaz a condição

$$\|f(x) - M_p(x)\|_\infty \leq \varepsilon,$$

com n um número inteiro e finito.

O cálculo dos parâmetros $c_0, c_1, c'_1, \ldots, c_n, c'_n$ do modelo (7.13) usa a técnica dos mínimos quadrados.

Aproximação do tipo séries de Fourier

Considere a *função distância dos mínimos quadrados*, $d(f, M_p)$, definida por:

$$d(f, M_p) = \sqrt{\int_{-\pi}^{\pi}(f(x) - M_p(x))^2\, dx}. \tag{7.14}$$

Se o produto escalar, entre as funções f e g, for definido por

$$\langle f(x), g(x)\rangle = \int_{-\pi}^{\pi} f(x)g(x)\, dx$$

para todas as funções f e g contínuas e periódicas, de período 2π, então as condições de ortogonalidade

$$\int_{-\pi}^{\pi} cos(jx)\, cos(kx)\, dx = 0,\ \ j \neq k,$$

$$\int_{-\pi}^{\pi} sen(jx)\, sen(kx)\, dx = 0,\ \ j \neq k$$

e

$$\int_{-\pi}^{\pi} cos(jx)\, sen(kx)\, dx = 0,$$

são verificadas, para todos os inteiros não negativos j e k.

Tem-se, então

Teorema 7.2 : O polinómio trigonométrico definido em (7.13) minimiza a função distância (7.14), se e só se os seus coeficientes tiverem os seguintes valores:

$$c_j = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\,cos(jx)\,dx, \quad j = 0, 1, \ldots, n \tag{7.15}$$

e

$$c'_j = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\,sen(jx)\,dx, \quad j = 1, \ldots, n. \tag{7.16}$$

O polinómio trigonométrico (7.13), com os coeficientes dados por (7.15) e (7.16), define a melhor aproximação dos mínimos quadrados de grau n, a $f(x)$.

Mesmo quando a função $f(x)$ pode ser calculada para qualquer valor de x, pode vir a ser necessário aproximar numericamente os integrais das fórmulas (7.15) e (7.16).

Os coeficientes em (7.15) e (7.16) satisfazem certas propriedades:

1. são independentes do valor de n;

2. verificam a condição

$$\frac{1}{2}c_0^2 + \sum_{j=1}^{n}((c_j)^2 + (c'_j)^2) \leq \frac{1}{\pi}\int_{-\pi}^{\pi}(f(x))^2\,dx,$$

conhecida por *desigualdade de Bessel.*

Como o lado direito da desigualdade não depende de n, pode-se concluir que

$$\frac{1}{2}c_0^2 + \sum_{j=1}^{\infty}((c_j)^2 + (c'_j)^2)$$

converge e as sequências $\{c_j, j = 0, 1, \ldots, n\}$ e $\{c'_j, j = 1, \ldots, n\}$ tendem para zero. A diferença entre os dois membros da desigualdade é uma medida da precisão da aproximação $M_p(x)$, em relação a $f(x)$, e é igual a $\|f(x) - M_p(x)\|_2^2$.

A periodicidade é uma propriedade natural do polinómio trigonométrico. Quando a função dada não é periódica, torna-se, por vezes, necessário aproximá-la por um polinómio trigonométrico, desde que diga respeito a um intervalo finito. A função dada pode considerar-se *estendida* periodicamente fora desse intervalo. Funções *pares*[7] e *impares*[8] são muitas vezes usadas como extensões.

Exemplo 7.7 Considere a seguinte função periódica definida por

$$f(x) = \begin{cases} -1 & \text{para } -\pi \leq x < 0 \\ 0 & \text{para } x = 0 \\ +1 & \text{para } 0 < x < \pi \end{cases}$$

[7]Uma função par tem a seguinte propriedade: $f(-x) = f(x)$. Como exemplo, tem-se a função $cos(x)$.
[8]Uma função impar tem a seguinte propriedade: $f(-x) = -f(x)$. Exemplo: $f(x) = sen(x)$.

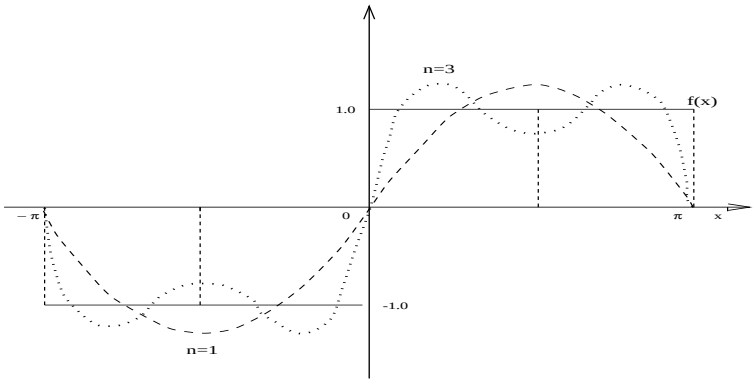


Figura 7.3: Aproximações a $f(x)$, com $n = 1$ e $n = 3$

com $f(x + 2\pi) = f(x)$.

Como é uma função impar, $c_j = 0$, $j = 0, 1, \ldots, n$ e

$$c'_j = \frac{2}{\pi} \int_0^{\pi} sen(jx)\, dx = \frac{4}{j\pi} \text{ se } j \text{ é impar e } 0 \text{ se } j \text{ é par.}$$

Na figura 7.3 estão representadas duas aproximações trigonométricas $M_p(x)$, a $f(x)$, que correspondem a $n = 1$ e $n = 3$.

Método discreto de Fourier

Quando não se conhece o valor de $f(x)$, para todo o $x \in [-\pi, \pi]$, mas apenas valores de f, para um conjunto discreto de m pontos, igualmente espaçados,

$$f(x_k) \text{ para } x_k = \frac{2\pi(k-1)}{m}, \quad k = 1, \ldots, m, \tag{7.17}$$

é, ainda possível, aproximar $f(x)$ por polinómios trigonométricos. Considera-se conhecido $f(0)$. No entanto, se forem dados os valores

$$f(\frac{2\pi(k-1)}{m} + a), \quad k = 1, \ldots, m,$$

para alguma constante $a \neq 0$, é possível fazer a mudança de variável do tipo $t = x - a$ e obter uma função $g(t) = f(t + a)$, para a qual se conhece $g(0)$.

A aproximação discreta do tipo séries de Fourier, à função f, é construída a partir dos dados definidos em (7.17). O polinómio trigonométrico, de grau n, tem a forma $M_p(x)$, definida em (7.13), e os seus coeficientes são calculados a partir de

$$c_j = \frac{2}{m} \sum_{k=1}^{m} f(\frac{2\pi(k-1)}{m})\, cos(\frac{2\pi j(k-1)}{m}), \quad j = 0, 1, \ldots, n \tag{7.18}$$

e

$$c'_j = \frac{2}{m}\sum_{k=1}^{m} f(\frac{2\pi(k-1)}{m})\ sen(\frac{2\pi j(k-1)}{m}),\ \ j = 1,\ldots,n. \tag{7.19}$$

Estas duas expressões foram obtidas a partir das equações, respectivamente, (7.15) e (7.16), substituindo os integrais por estimativas da forma

$$\frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\,dx \approx \frac{2}{m}\sum_{k=1}^{m} f(\frac{2\pi(k-1)}{m}).$$

Idênticas expressões para os coeficientes c_j, $j = 0, 1, \ldots, n$ e c'_j, $j = 1, \ldots, n$ seriam obtidas se a aproximação trigonométrica fosse calculada com base na minimização da soma dos quadrados

$$\sum_{k=1}^{m}[f(\frac{2\pi(k-1)}{m}) - M_p(\frac{2\pi(k-1)}{m})]^2$$

com $M_p(x)$ definida por (7.13). É de realçar, que a condição $n < \frac{m}{2}$ $(2n+1 \leq m)$ deve ser verificada para que este problema de mínimos quadrados tenha uma única solução.

Exemplo 7.8 Para o seguinte conjunto de dados

x_i	0	$\pi/2$	π	$3\pi/2$
$f(x_i)$	0.2	0.25	1.0	0.5

a aproximação, do tipo $M_p(x) = \frac{1}{2}c_0 + c_1 cos(x) + c'_1 sen(x)$, de grau um, é obtida usando o método discreto de Fourier. Assim, para $m = 4$, $n = 1$, tem-se

$$c_0 = 0.975,\ \ c_1 = -0.4 \text{ e } c'_1 = -0.125,$$

e a desigualdade de Bessel verifica-se: $0.8896 \leq 2.1245$.

7.3 Problema de mínimos quadrados não linear

7.3.1 Introdução teórica

Seja $M : \mathbf{R}^z \times \mathbf{R}^n \to \mathbf{R}$ uma função dada de z variáveis reais (possivelmente de uma só) $x_1, x_2, \ldots, x_z$ e n parâmetros reais (desconhecidos) $c_1, c_2, c_3, \ldots, c_n$ e suponha que a variável f depende das variáveis x_j, $j = 1, \ldots, z$, através da relação

$$f = M(x_1, \ldots, x_z; c_1, c_2, \ldots, c_n), \tag{7.20}$$

que é *não linear* nas variáveis $c_1, c_2, \ldots, c_n$. Estas n variáveis formam o vector c de $\mathbf{R}^n$.

Se um conjunto de medições de f forem feitas correspondendo ao mesmo número de valores de x_j:

$$(x_{11}, \ldots, x_{z1}, f_1), (x_{12}, \ldots, x_{z2}, f_2), \ldots, (x_{1m}, \ldots, x_{zm}, f_m),$$

então o modelo (7.20) pode ser ajustado aos dados experimentais de tal forma que os valores de $c_1, c_2, \ldots, c_n$, que asseguram o 'melhor ajuste', possam ser determinados.

Tomando $m \geq n$ (de preferência $m > n$) e definindo a função $S : \mathbf{R}^n \to \mathbf{R}$ por

$$S(c_1, \ldots, c_n) = \sum_{j=1}^{m} [f_j - M(x_{1j}, \ldots, x_{zj}; c_1, \ldots, c_n)]^2,$$

o ajuste do modelo (7.20) aos dados experimentais $f_j, j = 1, \ldots, m$, no sentido dos mínimos quadrados, corresponde à *minimização da função* $S(c_1, \ldots, c_n)$,

$$\min_{c \in \mathbf{R}^n} S(c_1, \ldots, c_n). \tag{7.21}$$

A solução $c^* = (c_1^*, c_2^*, \ldots, c_n^*)^T$ do problema (7.21) ajusta os dados experimentais no sentido dos mínimos quadrados.

Como o objectivo é calcular o mínimo local da função $S(c)$, que é não linear nas variáveis $c_1, \ldots, c_n$, uma condição necessária para que c^* seja um mínimo local de $S(c)$ (veja-se no Capítulo 10 em 10.1.4.) é a de que

$$\frac{\partial S(c_1^*, \ldots, c_n^*)}{\partial c_i} = 0\,, \quad \text{para } i = 1, \ldots, n.$$

Esta condição corresponde ao seguinte *sistema das equações normais*

$$F_i(c_1, \ldots, c_n) = -\sum_{j=1}^{m} [f_j - M(x_{1j}, \ldots, x_{zj}; c_1, c_2, \ldots, c_n)] \frac{\partial M(x_j; c)}{\partial c_i} = 0, \tag{7.22}$$

$$\text{para } i = 1, \ldots, n$$

e que se apresenta como não linear nas variáveis $c_1, c_2, \ldots, c_n$.

Os métodos para resolver o problema (7.21) são iterativos e calculam uma solução do sistema não linear (7.22), $F(c) = 0$. $F : \mathbf{R}^n \to \mathbf{R}^n$ é o vector formado pelas funções $F_1, F_2, \ldots, F_n$ definidas por (7.22) e $c \in \mathbf{R}^n$ é o vector dos parâmetros a determinar. Esta solução define um ponto estacionário da função S, que pode não ser um mínimo.

7.3.2 Método de Newton

Se for possível calcular a matriz do Jacobiano do sistema (7.22), que representamos por

$$A(c_1, \ldots, c_n) = (a_{ik}), \text{ com } i, k = 1, \ldots, n,$$

e cujos elementos são definidos por

$$a_{ik} = \frac{\partial F_i}{\partial c_k} = \sum_{j=1}^{m} \{ \frac{\partial M(x_j; c)}{\partial c_k} \frac{\partial M(x_j; c)}{\partial c_i} - [f_j - M(x_j; c)] \frac{\partial^2 M(x_j; c)}{\partial c_k \partial c_i} \}\,, \tag{7.23}$$

para $i, k = 1, \ldots, n$, então pode-se usar o *método de Newton* (veja-se em 4.2 no Capítulo 4) desde que uma boa aproximação à solução seja conhecida. O método de Newton gera uma sequência de aproximações $\{c^{(l)}\}$,

$$c^{(l+1)} = c^{(l)} + \Delta_c$$

a partir da equação matricial iterativa

$$A(c^{(l)})\Delta_c = -F(c^{(l)}), \quad l = 1, 2, \ldots. \tag{7.24}$$

em que $A(c^{(l)})$ designa a matriz do Jacobiano (cujos elementos foram definidos em (7.23)) calculada no ponto $c^{(l)}$, da iteração l, e $F(c^{(l)})$ é o vector $(F_1, F_2, \ldots, F_n)^T$ (com elementos definidos por (7.22)), calculado no ponto $c^{(l)}$.

A implementação do método de Newton envolve o cálculo

- das mn primeiras derivadas parciais do modelo, $\frac{\partial M(x_j;c)}{\partial c_i}$ $(i = 1, \ldots, n; j = 1, \ldots, m)$, para construir o vector F, em (7.22), e parte da matriz A, em (7.23), e
- das $mn\frac{(n+1)}{2}$ segundas derivadas parciais do modelo, $\frac{\partial^2 M(x_j;c)}{\partial c_k \partial c_i}$ $(i, k = 1, \ldots, n; j = 1, \ldots, m)$, para a matriz do Jacobiano A, em (7.23).

O algoritmo que corresponde ao método de Newton para a resolução do problema (7.21) é o algoritmo 7.3.

Algoritmo 7.3 :

1. ler m, n, z, n_{max} e para $j = 1, \ldots, m$ ler f_j e para $l = 1, \ldots, z$ ler $x_{l\,j}$
2. introduzir a função $M(x_1, \ldots, x_z; c_1, c_2, \ldots, c_n)$
3. introduzir as primeiras derivadas, $\frac{\partial M(x;c)}{\partial c_i}$ para $i = 1, \ldots, n$ e as segundas, $\frac{\partial^2 M(x;c)}{\partial c_i \partial c_k}$ para $i = 1, \ldots, n$ e $k = i, \ldots, n$
4. ler a aproximação inicial: para $i = 1, \ldots, n$ ler $c_i^{(1)}$ e fazer $l = 0$
5. fazer $l = l + 1$ e para $j = 1, \ldots, m$

   5.1. calcular $M_j = M(x_{1j}, \ldots, x_{zj}; c_1^{(l)}, \ldots, c_n^{(l)})$

   5.2. para $i = 1, \ldots, n$ calcular $der_{ji} = \frac{\partial M(x_{1j}, \ldots, x_{zj}; c_1^{(l)}, \ldots, c_n^{(l)})}{\partial c_i}$
6. calcular $S(c^{(l)}) = \sum_{j=1}^{m}(f_j - M_j)^2$
7. calcular o vector F: para $i = 1, \ldots, n$ calcular $F_i = -\sum_{j=1}^{m}[f_j - M_j]der_{ji}$ e fazer $b_i = -F_i$
8. se $l \geq 2$ então

   8.1. se convergência=.TRUE. (algoritmo 4.1) então ir para o passo 12

   8.2. se $l \geq n_{max}$ então recomeçar, ir para o passo 4

9. calcular a matriz do Jacobiano A:

   9.1. para $j = 1, \ldots, m$, $i = 1, \ldots, n$ e $k = i, \ldots, n$ calcular
   $seg_{ikj} = \frac{\partial^2 M(x_{1j}, \ldots, x_{zj}; c_1^{(l)}, \ldots, c_n^{(l)})}{\partial c_i \partial c_k}$

   9.2. para $i = 1, \ldots, n$ e $k = i, \ldots, n$

      9.2.1. calcular $a_{ik} = \sum_{j=1}^{m} [der_{ji} der_{jk} - (f_j - M_j) seg_{ikj}]$

      9.2.2. se $k \neq i$ então fazer $a_{ki} = a_{ik}$

10. resolver o sistema das n equações lineares (algoritmo 3.3) $Ax = b$, $(x \in \mathbb{R}^n)$ e colocar a solução em Δ_{c_i}, para $i = 1, \ldots, n$

11. para $i = 1, \ldots, n$ calcular $c_i^{(l+1)} = c_i^{(l)} + \Delta_{c_i}$ e ir para o passo 5

12. terminar com $c^* \leftarrow (c_i^{(l+1)}, i = 1, \ldots, n)$.

7.3.3 Implementação. Rotina MQNEWT

O algoritmo 7.3 está implementado na rotina MQNEWT. Os valores atribuídos aos parâmetros ε_1 e ε_2 do critério de paragem (algoritmo 4.1) deste processo iterativo são 10^{-6}. O critério exige que as duas condições sejam verificadas:

$$\|F(c^{(l+1)})\|_2 \leq \varepsilon_2$$

com F definido em (7.22) e

$$(\|\Delta_c\|_2 / \|c^{(l+1)}\|_2) \leq \varepsilon_1$$

com Δ_c calculado de (7.24).

Exemplo 7.9 Pretende-se minimizar

$$\sum_{j=1}^{3} [f_j - c_1(1 - c_2^{x_j})]^2$$

sabendo que os valores de f são os da tabela

x_j	1	2	3
f_j	1.5	2.25	2.625

Tem-se um conjunto de três observações para determinar dois parâmetros c_1 e c_2. O modelo que se pretende ajustar é $M(x; c_1, c_2) = c_1(1 - c_2^x)$ e os valores iniciais atribuídos são $c^{(1)} = (0.1, 0.1)^T$. Na primeira iteração, o valor de $S = 12.991031$, $F = (-5.921065, 0.191595)^T$ e a matriz do Jacobiano é

$$A = \begin{pmatrix} 2.788101 & 1.803156 \\ 0.803156 & 0.592115 \end{pmatrix}.$$

A rotina MQNEWT ao fim de 100 iterações ainda não tinha convergido e o valor de S ainda é ≈ 0.45. Neste exemplo, espera-se atingir um valor de $S = 0$. Compare os resultados com os do exemplo 7.11. Este comportamento é justificado pelo facto da matriz do Jacobiano (7.23), do

sistema Newton, não ser definida positiva, ao longo de todo o processo, e as direcções Δ_c não serão provavelmente de descida em relação a $S(c)$.

Exemplo 7.10 Dada a função $f(x)$ definida pela tabela de seis valores

x_j	-5	-3	-1	1	3	5
f_j	127	151	379	421	460	426

e o modelo $M(x; c_1, c_2, c_3) = c_1 + c_2\, e^{x\, c_3}$ pretende-se calcular os parâmetros c_1, c_2 e c_3 que ajustam o modelo dado aos valores da função no sentido dos mínimos quadrados. Iniciando o processo iterativo com $c^{(1)} = (580, -180, -0.16)^T$, a rotina MQNEWT fornece a solução $c^* = (523.305539, -156.947844, -0.199665)^T$ ao fim de 8 iterações. O valor de S atingido é 13390.093119 e $F_1 = F_2 = F_3 = 0.0000000$. Este é um dos exemplos em que S não atinge o valor zero; é um problema de grandes resíduos.

7.3.4 Aproximação Gauss-Newton

Se $S(c^*)$ chega a atingir um valor muito próximo de (ou igual a) zero, em virtude de

$$S(c) = [f - M(x; c)]^T [f - M(x; c)]\,,$$

as diferenças $f_j - M(x_{1j}, \ldots, x_{zj}; c_1^*, c_2^*, \ldots, c_n^*)$, para $j = 1, \ldots, m$ são também muito pequenas (ou nulas). Assim, quando a sequência de aproximações $\{c^{(l)}\}, l = 1, 2, \ldots$ se aproxima de c^*, o segundo termo na definição da matriz do Jacobiano, $A(c)$, definida em (7.23), pode ser desprezado, desde que as segundas derivadas de $M(x; c)$ sejam funções limitadas. Nestas condições pode-se aproximar a matriz do Jacobiano através de

$$a_{ik} = \frac{\partial F_i}{\partial c_k} \approx \sum_{j=1}^{m} \frac{\partial M(x_j; c)}{\partial c_k} \frac{\partial M(x_j; c)}{\partial c_i}\,, \quad \text{para} \quad i, k = 1, \ldots, n \tag{7.25}$$

e o método de Newton para a resolução do problema de mínimos quadrados não linear dá origem a uma implementação bastante mais económica e simples, conhecida por *implementação Gauss-Newton*. Esta implementação apenas exige o cálculo das primeiras derivadas parciais do modelo $M(x; c)$.

Uma das maiores dificuldades, na implementação dos métodos de Newton e Gauss-Newton, surge quando a aproximação inicial $c^{(1)}$ não se encontra suficientemente perto da solução, não sendo possível garantir convergência da sequência $\{c^{(l)}\}$ para c^*.

Se a matriz do Jacobiano (no método de Newton), ou uma aproximação (no método de Gauss-Newton), identificada pela matriz $A = (a_{ik}), i, k = 1, \ldots, n$, for definida positiva, então o vector Δ_c, que se calcula a partir da resolução do sistema linear

$$A(c^{(l)})\, \Delta_c = -F(c^{(l)})\,,$$

verifica a condição

$$\Delta_c^T F(c^{(l)}) < 0, \tag{7.26}$$

o que significa que o vector Δ_c é de *descida* em relação à função $S(c)$, no ponto $c^{(l)}$. Nestas condições, existe um escalar $\alpha > 0$, que dá o comprimento do deslocamento a afectuar ao longo da direcção Δ_c, de tal forma que

$$S(c^{(l)} + \alpha\,\Delta_c) < S(c^{(l)}).$$

Esta relação significa que, a partir do ponto $c^{(l)}$, pode-se caminhar ao longo de Δ_c e encontrar um novo ponto, $c^{(l)} + \alpha\,\Delta_c$, para o qual a função S toma um valor mais baixo do que em $c^{(l)}$. Por razões que explicaremos mais adiante, no Capítulo 10, é razoável escolher o escalar α por forma a que se verifique, em cada iteração, não apenas um decréscimo de S, mas uma redução considerada significativa de S, isto é,

$$S(c^{(l)} + \alpha\,\Delta_c) \leq S(c^{(l)}) + 2\,\beta\,\alpha\,\Delta_c^T F(c^{(l)}) \tag{7.27}$$

em que $\beta > 0$ é um parâmetro fixo e dado, definido em $(0, 1)$.

Uma estratégia simples para o cálculo do valor de $\alpha \in [0, 1]$, que satisfaz a condição (7.27) está representada *nos passos 11., 12., 13., 14., 15. e 16. do algoritmo 7.4.* Corresponde a dividir o valor de α sucessivamente por dois, até se encontrar o primeiro valor que verifique (7.27). O valor inicial para começar as sucessivas divisões é igual a um.

A *implementação de Hartley*, nome por que é conhecido o algoritmo de Gauss-Newton com este procedimento de procura do valor de α, está representada no algoritmo 7.4.

O valor atribuído ao parâmetro β pode influenciar a implementação. Para valores mais próximos de zero, por exemplo, 10^{-3}, tem-se uma redução menor dos valores de S, de iteração para iteração. Para valores maiores de β, por exemplo, $\beta = 0.5$, a redução exigida na função S é muito significativa.

Na implementação do algoritmo 7.3 também se pode usar este procedimento da escolha do escalar α. Embora, mesmo assim, não haja garantia de convergência, a prática mostra que a implementação é mais robusta, isto é, é bem sucedida mais vezes. Na rotina MQNEWT foi introduzido este esquema.

Algoritmo 7.4 :

1. ler m, n, z, n_{max}, β e para $j = 1, \ldots, m$ ler f_j e para $l = 1, \ldots, z$ ler $x_{l\,j}$
2. introduzir a função $M(x_1, \ldots, x_z; c_1, c_2, \ldots, c_n)$
3. introduzir as primeiras derivadas, $\frac{\partial M(x;c)}{\partial c_i}$, para $i = 1, \ldots, n$
4. ler a aproximação inicial: para $i = 1, \ldots, n$ ler $c_i^{(1)}$ e fazer $l = 0$
5. fazer $l = l + 1$ e para $j = 1, \ldots, m$
   - 5.1. calcular $M_j = M(x_{1j}, \ldots, x_{zj}; c_1^{(l)}, \ldots, c_n^{(l)})$
   - 5.2. para $i = 1, \ldots, n$ calcular $der_{ji} = \frac{\partial M(x_{1j}, \ldots, x_{zj}; c_1^{(l)}, \ldots, c_n^{(l)})}{\partial c_i}$
6. calcular $S(c^{(l)}) = \sum_{j=1}^{m} [f_j - M_j]^2$

7. calcular o vector F: para $i = 1, \ldots, n$ calcular $F_i = -\sum_{j=1}^{m}[f_j - M_j]der_{ji}$ e fazer $b_i = -F_i$
8. se $l \geq 2$ então
   - 8.1. se convergência=.TRUE. (algoritmo 4.1) então ir para o passo 18
   - 8.2. se $l \geq n_{max}$ então recomeçar, ir para o passo 4
9. calcular uma aproximação ao Jacobiano A: para $i = 1, \ldots, n$ e $k = i, \ldots, n$
   - 9.1. calcular $a_{ik} = \sum_{j=1}^{m}[der_{ji}der_{jk}]$
   - 9.2. se $k \neq i$ então fazer $a_{ki} = a_{ik}$
10. resolver o sistema das n equações lineares (algoritmo 3.3) $Ax = b$, $(x \in \mathbb{R}^n)$ e colocar a solução em Δ_{c_i}, para $i = 1, \ldots, n$
11. (determinação do escalar α) fazer $\alpha = 1$
12. calcular $prod = 2\sum_{i=1}^{n} \Delta_{c_i} F_i$
13. para $i = 1, \ldots, n$ calcular $c_i^{(l+1)} = c_i^{(l)} + \alpha\, \Delta_{c_i}$
14. para $j = 1, \ldots, m$ calcular $M_j = M(x_{1j}, \ldots, x_{zj}; c_1^{(l+1)}, \ldots, c_n^{(l+1)})$
15. calcular $S(c^{(l+1)}) = \sum_{j=1}^{m}[f_j - M_j]^2$
16. se $S(c^{(l+1)}) > S(c^{(l)}) + \beta\, \alpha\, prod$ então fazer $\alpha = \frac{\alpha}{2}$ e ir para o passo 13
17. ir para o passo 5
18. terminar com $c^* \leftarrow (c_i^{(l+1)}, i = 1, \ldots, n)$.

7.3.5 Implementação. Rotina MQGANE

O algoritmo 7.4 foi implementado para resolver os exemplos 7.9 e 7.10, através da rotina MQGANE. Os parâmetros ε_1 e ε_2 do critério de paragem continuam a ser iguais a 10^{-6} e $\beta = 10^{-3}$.

Exemplo 7.11 Considere o problema do exemplo 7.9. Como este é um dos exemplos em que S atinge um valor muito próximo de zero, a rotina MQGANE dá resultados satisfatórios. Para este problema, embora MQNEWT não tenha convergido, obtém-se com a rotina MQGANE convergência ao fim de 7 iterações, para o mesmo vector inicial $c^{(1)} = (0.1, 0.1)^T$. O valor calculado para c^* é $(3.0, 0.5)^T$ e $S^* = 0.000000$.

Exemplo 7.12 Para o problema do exemplo 7.10 a rotina MQGANE converge ao fim de 37 iterações. Tomando o vector inicial de 7.10, $(580, -180, -0.16)^T$, em que $S = 27376.618648$ e $F = (267.631178, 370.780056, 190012.506115)^T$, chega-se a $c^* = (523.305539, -156.947844, -0.199665)^T$ com

$$F = (-0.000000, -0.000000, 0.000043)^T \text{ e } S^* = 13390.093119.$$

Aumentando ε_1 e ε_2 para 10^{-4}, MQGANE pára ao fim de 28 iterações, com

$$(523.305538, -156.947843, -0.199665)^T.$$

7.4 Problemas

1. Para a função $f(x)$ definida pela tabela de oito valores

x_i	0.1	0.2	0.3	0.5	0.7	1.0	1.4	2.0
f_i	1.97	3.81	5.40	7.85	9.51	11.1	12.3	13.3

ajuste, no sentido dos mínimos quadrados, os seguintes modelos:

i) $p_1(x) = c_0 P_0(x) + c_1 P_1(x)$, com $P_0(x)$ e $P_1(x)$ polinómios ortogonais;

ii) polinómio de grau 2, $p_2(x)$;

iii) $M(x; c_1, c_2) = c_1 e^{c_2 x}$.

Compare os valores obtidos para $p_1(0.25)$, $p_2(0.25)$ e $M(0.25)$. Represente graficamente estas três funções, bem como os valores da tabela. Que conclusões pode tirar destes resultados?

2. Pretende-se ajustar o modelo

$$M(x; c_1, c_2) = c_1 x + c_2\, sen(x)$$

à função $f(x)$ dada pela tabela:

x_i	0	0.5	1.4
$f(x_i)$	0	3	4

no sentido dos mínimos quadrados. Determine os coeficientes do modelo apresentado.

3. Considere a tabela de valores da função $f(x)$:

x_i	0.0	0.1	0.5
$f(x_i)$	2	1	2

Calcule o modelo

$$M(x; c_1, c_2) = c_1 + c_2 e^{-x}$$

que melhor ajusta à função, no sentido dos mínimos quadrados.

Dê uma estimativa de $f(0.4)$.

4. Calcule a solução do problema

$$\min_c \sum_{i=1}^{10} [f_i - M(x_i; c)]^2$$

a partir da tabela de valores de f

x_i	0	1	2	3	4	6	8	10	15	20
f_i	4.0	4.7	4.9	5.3	6.1	6.7	6.9	7.2	7.1	7.5

para os seguintes casos:

i) $M(x;c) = c_0P_0(x) + c_1P_1(x) + c_2P_2(x)$, com $P_0(x), P_1(x)$ e $P_2(x)$ polinómios ortogonais e $c = (c_0, c_1, c_2)^T$;

ii) $M(x;c) = c_1 + c_2x + \frac{c_3}{x}$, com $c = (c_1, c_2, c_3)^T$;

iii) $M(x;c) = c_1x + c_2e^x$, com $c = (c_1, c_2)^T$.

Calcule aproximações a $f(5)$, usando os três modelos de funções de aproximação a $f(x)$. Qual dos casos parece apresentar uma aproximação mais razoável, pelo menos no intervalo $[4, 6]$.

5. Seja $f(x)$ uma função contínua, par e periódica de período 2π, definida no intervalo $[-\pi, \pi]$ por $f(x) = |x|$ e que foi estendida periodicamente. Verifique que $c_0 = \pi$,

$$c_j = \frac{2}{\pi}\int_0^{\pi} x\,cos(jx)\,dx = \frac{2(cos(j\pi) - 1)}{\pi j^2}$$

e $c'_j = 0$, para qualquer j.

Construa o polinómio trigonométrico de grau 1, que melhor aproxima a função dada, no sentido dos mínimos quadrados.

6. Considere a função contínua definida no intervalo $[-\pi, \pi]$ por

$$f(x) = \begin{cases} x(\pi + x) & \text{para } -\pi \leq x \leq 0 \\ x(\pi - x) & \text{para } 0 \leq x \leq \pi \end{cases}$$

e que foi estendida como uma função impar de período 2π. Verifique que $c_j = 0$ para $j = 0, 1, \ldots, n$ e

$$c'_j = \frac{4(1 - cos(j\pi))}{\pi j^3},\ j = 1, \ldots, n$$

Construa o polinómio trigonométrico de grau 2, que melhor aproxima a função dada, no sentido dos mínimos quadrados.

7. Considere a função impar definida por

$$f(x) = 1 - \frac{4}{\pi^2}(x - \frac{1}{2}\pi)^2, \quad 0 \leq x \leq \pi.$$

Calcule uma aproximação do tipo séries de Fourier a $f(x)$.

8. Dados os seguintes valores da função $f(x)$: $f(0) = 0.2$, $f(\frac{1}{2}\pi) = 0.25, f(\pi) = 1.0$ e $f(1\frac{1}{2}\pi) = 0.5$, use o método discreto de Fourier para calcular o polinómio

$$M_p(x) = \frac{1}{2}c_0 + c_1 cos(x) + c_1' sen(x) + c_2 cos(2x), \quad -\infty < x < \infty$$

para aproximar f.

9. Seja $f(x)$ uma função contínua definida no intervalo $[-\pi, \pi]$ por $f(x) = x^2$ e que foi estendida periodicamente. Determine as aproximações do tipo séries de Fourier,

$$M_p(x) = \frac{1}{2}c_0 + c_1 cos(x) + c_1' sen(x), \quad -\infty < x < \infty$$

$$M_p'(x) = \frac{1}{2}c_0 + c_1 cos(x) + c_1' sen(x) + c_2 cos(2x) + c_2' sen(2x), \quad -\infty < x < \infty$$

à função f, baseadas nos seguintes pontos $x_k = \frac{2\pi(k-1)}{6}$, para $k = 1, \ldots, 6$.

Qual delas parece ser a melhor aproximação? Justifique.

10. Implemente o método de Gauss-Newton, com o objectivo de ajustar, o melhor possível, o modelo

$$M(x; c_1, c_2) = c_1 sen(c_2 x)$$

à função $f(x)$ dada pela tabela

x_i	0	0.5	1
$f(x_i)$	0	1.25	1.35

no sentido dos mínimos quadrados. Como vector inicial use $c^{(1)} = (2, 2)^T$, tome $\beta = 0.001$ e $\varepsilon_1 = \varepsilon_2 = 0.001$. Verifique que obtém para $(c_1^*, c_2^*) = (4.496521, 1.460840)$ e $S^* = 0.000000$.

11. Implemente o método de Gauss-Newton, com o objectivo de ajustar o melhor possível o modelo

$$M(x, c_1, c_2) = c_1 sen(x) + cos(c_2 x)$$

à função $f(x)$ dada pela tabela:

x_i	-1	0	1
$f(x_i)$	-1.1	1.0	2.2

no sentido dos mínimos quadrados. Como aproximações iniciais aos parâmetros use $(2, 1)^T$.

Use no critério de paragem $\varepsilon_1 = \varepsilon_2 = 0.1$.

12. Implemente o método de Gauss-Newton, com o objectivo de ajustar o melhor possível o modelo

$$M(x; c_1, c_2) = c_1 + sen(c_2 x)$$

à função $f(x)$ dada pela seguinte tabela de três pontos:

x_i	-1	0	1
$f(x_i)$	0.9	1	1.1

no sentido dos mínimos quadrados. Como aproximações iniciais aos parâmetros use $(1, 2)^T$. Pare o processo iterativo quando o critério de paragem for verificado para $\varepsilon_1 = 0.1$ e $\varepsilon_2 = 0.1$.

13. Implemente os métodos de Newton e Gauss-Newton, com o objectivo de ajustar o melhor possível o modelo

$$M(x; c_1, c_2) = c_1 + c_1 c_2 x$$

à função $f(x)$ dada pela seguinte tabela de três pontos:

x_i	-1	0	1
$f(x_i)$	1	2	3

no sentido dos mínimos quadrados. Como aproximações iniciais aos parâmetros use $(1.5, 0.75)^T$. Pare os processos iterativos quando o critério de paragem for verificado para $\varepsilon_1 = \varepsilon_2 = 0.5$.

14. Dados a função $f(x_1, x_2)$ definida pela seguinte tabela de cinco valores

x_{1i}	1.0	2.0	1.0	2.0	0.1
x_{2i}	1.0	1.0	2.0	2.0	0.0
f_i	0.126	0.219	0.076	0.126	0.186

e o modelo

$$M(x_1, x_2; c_1, c_2, c_3) = \frac{c_1 c_3 x_1}{1 + c_1 x_1 + c_2 x_2},$$

calcule valores para os parâmetros c_1, c_2 e c_3 que optimizam o ajuste dos valores da tabela ao modelo dado, no sentido dos mínimos quadrados. Considere para aproximação inicial o vector $c^{(1)} = (10.39, 48.83, 0.74)^T$ e para os parâmetros do critério de paragem ε_1 e ε_2, o valor 10^{-6}.

Verifique que a implementação do algoritmo Gauss-Newton, conhecida por implementação de Hartley converge para a solução $c^* = (3.131505, 15.159362, 0.780063)^T$ ao fim de 5 iterações. O valor de S conseguido é 0.000044.

Resolva novamente o problema usando, desta vez, a implementação Newton e introduzindo o procedimento da escolha do escalar α. Verifique que o processo não progride a partir da 4^a iteração e a condição (7.27) obriga a uma divisão sucessiva de α por dois, quando $\beta = 10^{-3}$. Diminua este valor um pouco mais e veja o que acontece. Que conclusões pode tirar desta resolução?

Capítulo 8

Integração numérica

8.1 Introdução

8.1.1 Forma geral do problema

Neste Capítulo interessa calcular uma aproximação ao integral

$$I = \int_a^b f(x)dx$$

com a e b constantes finitas. A computação numérica de integrais definidos é um dos problemas mais antigos em matemática[1]. A *integração numérica* é também conhecida pelo nome de *quadratura numérica*. Esta designação deriva do facto de se ter tentado calcular a área de um círculo, através da determinação de um quadrado com a mesma área.

8.1.2 Características do problema

Três das situações mais comuns, onde se torna necessário calcular uma aproximação ao integral definido, são:

- a primeira, inclui os casos em que a primitiva da função integranda não pode vir expressa em termos de funções elementares;
- a segunda, envolve funções integrandas com primitivas exactas, mas cuja expressão é tão complicada que uma aproximação numérica torna-se desejável;
- finalmente a terceira surge quando a função integranda é conhecida apenas para um conjunto discretos de pontos.

[1] À integração numérica estão associados nomes célebres como Newton, Stirling, Cotes, Gregory, Simpson, Wallis, o primeiro a utilizar o símbolo ∞ para designar infinito, num trabalho sobre integração, Euler e MacLaurin, que descobriram independentemente um do outro, este em 1737 e aquele em 1730, o teorema que estabelece um dos resultados mais importantes em integração numérica, a fórmula de Euler-MacLaurin. Falta mencionar Cavalieri que em 1639 foi um dos pioneiros no domínio da integração (Goldstine (1977)).

8.1.3 Tipos de métodos

O problema da integração numérica está fortemente relacionado com a resolução de equações diferenciais (veja-se o Capítulo 9) e as técnicas usadas baseiam-se em interpolação. Assim, em vez de se resolver directamente $I = \int_a^b f(x)dx$, determina-se o valor de $f(x)$ para um conjunto seleccionado de pontos x_i, $i = 0, \ldots, n$ do intervalo $[a, b]$, calcula-se o polinómio interpolador, $p_n(x)$, baseado nos pontos $(x_i, f(x_i))$ e finalmente, usa-se $\int_a^b p_n(x)dx$ para aproximar o valor de I. Os métodos de integração numérica diferem entre si na maneira como seleccionam os pontos, no número de pontos utilizados e na maneira como estes são usados para construir o polinómio interpolador. Alguns métodos dividem o intervalo $[a, b]$ em subintervalos e aproximam a função por polinómios em cada um desses subintervalos, somando, no final, todas as contribuições. Outros utilizam pontos de $[a, b]$ igualmente espaçados. Outros, ainda, têm em consideração o número limitado de pontos onde a função é conhecida.

8.1.4 Índice de algoritmos

São apresentados três algoritmos neste Capítulo.

Algoritmo 8.1 Cálculo de um integral definido utilizando apenas uma fórmula, ou uma combinação das fórmulas composta de integração

Algoritmo 8.2 Cálculo de um integral definido utilizando a integração de Romberg combinada com a fórmula do trapézio

Algoritmo 8.3 Cálculo de um integral definido ou indefinido através das fórmulas Gaussianas

A utilização de algumas fórmulas composta de integração, definitivamente as mais simples e conhecidas, para intervalos, entre pontos, de amplitudes constantes e também para intervalos de amplitudes variadas, é feita no algoritmo 8.1. Resultados obtidos pela fórmula composta do trapézio podem ser melhorados introduzindo a técnica de Romberg, como é implementada no algoritmo 8.2. No algoritmo 8.3 são implementadas algumas fórmulas de Quadratura Gaussiana.

8.2 Fórmulas com intervalos de amplitudes constantes

8.2.1 Fórmulas de Newton-Cotes

Uma aproximação ao integral $I = \int_a^b f(x)dx$, calculada por uma das regras de integração, apresenta o aspecto de uma combinação linear de valores da função integranda $f(x)$, nos pontos $x_i, i = 0, \ldots, n$,

$$Q(f) = w_0 f(x_0) + w_1 f(x_1) + \cdots + w_n f(x_n).$$

A dedução destas regras reduz-se à aproximação de $f(x)$ por um polinómio de grau n, $p_n(x)$, que passa pelos pontos interpoladores $x_i, i = 0, \ldots, n$, e à integração exacta de

$p_n(x)$. Seleccionando o valor de n e os pontos interpoladores obtêm-se as diversas regras, hoje conhecidas por *fórmulas de Newton-Cotes*[2].

O polinómio interpolador, utilizando a fórmula interpoladora de Lagrange, tem a forma

$$p_n(x) = \sum_{i=0}^{n} L_i(x) f(x_i)$$

com o coeficiente de ordem i definido por

$$L_i(x) = \frac{(x-x_0)\dots(x-x_{i-1})(x-x_{i+1})\dots(x-x_n)}{(x_i-x_0)\dots(x_i-x_{i-1})(x_i-x_{i+1})\dots(x_i-x_n)}\,.$$

No cálculo de $\int_a^b p_n(x)dx$, ou

$$\int_a^b \sum_{i=0}^{n} L_i(x) f(x_i) dx$$

resulta

$$\sum_{i=0}^{n} w_i f(x_i)\,, \quad \text{com} \quad w_i = \int_a^b L_i(x)dx \tag{8.1}$$

donde se conclui que os coeficientes w_i dependem directamente da escolha dos pontos x_i e do seu número. No entanto, não dependem da função $f(x)$.

Considerem-se os pontos x_i, $i = 0, \dots, n$, pertencentes a $[a, b]$, com *espaçamento constante*, de tal forma que $x_i = a + ih\,, \quad i = 0, \dots, n$, com $h = \frac{b-a}{n}$.

As cinco regras de integração mais conhecidas podem ser deduzidas da seguinte maneira:

- se $n = 0$ e $x_0 = a$, tem-se $L_0 = 1$, $w_0 = \int_a^b dx = b - a$ (de (8.1)) e

  $$\int_a^b f(x)dx \approx (b-a) f(a)$$

  que dá pelo nome de *regra do rectângulo*[3];

- se $n = 0$, mas $x_0 = \frac{a+b}{2}$, tem-se $L_0 = 1\,,\ w_0 = \int_a^b dx = b - a$ e

  $$\int_a^b f(x)dx \approx (b-a) f(\frac{a+b}{2})$$

  conhecida por *regra do ponto médio*;

[2] R. Cotes viveu entre 1682 e 1716 e embora seja conhecido pelo trabalho que desenvolveu em integração, também esteve ligado à interpolação. Grande parte do seu trabalho foi publicada após a sua morte e editada por R. Smith, em 1722, e contém as regras de integração de Newton-Cotes, correspondentes a $n = 3, \dots, 11$ bem como alguns textos importantes sobre trigonometria (Goldstine (1977) e Hämmerlin e Hoffmann (1991)).

[3] A área do rectângulo de base $b-a$ e altura $f(a)$ é usada para aproximar a área definida por $\int_a^b f(x)dx$.

- se $n = 1$, com $x_0 = a$ e $x_1 = b$, então $L_0 = \frac{x-b}{a-b}$, $w_0 = \int_a^b L_0\, dx = \frac{b-a}{2}$, $L_1 = \frac{x-a}{b-a}$, $w_1 = \int_a^b \frac{x-a}{b-a}\, dx = \frac{b-a}{2}$ e

$$\int_a^b f(x)dx \approx \frac{b-a}{2}[f(a) + f(b)]$$

conhecida pelo nome de *regra do trapézio*[4];

- se $n = 2$, $x_0 = a$, $x_1 = \frac{a+b}{2}$ e $x_2 = b$, tem-se $w_0 = \int_a^b \frac{(x-\frac{a+b}{2})(x-b)}{(a-\frac{a+b}{2})(a-b)}\, dx = \frac{b-a}{6}$, $w_1 = \int_a^b \frac{(x-a)(x-b)}{(\frac{a+b}{2}-a)(\frac{a+b}{2}-b)}\, dx = \frac{4(b-a)}{6}$, $w_2 = \frac{b-a}{6}$ e

$$\int_a^b f(x)dx \approx \frac{b-a}{6}[f(a) + 4f(\frac{a+b}{2}) + f(b)]$$

que é a *regra de Simpson*[5];

- se $n = 3$, $x_0 = a$, $x_1 = \frac{2a+b}{3}$, $x_2 = \frac{a+2b}{3}$ e $x_3 = b$, tem-se $w_0 = \frac{b-a}{8} = w_3$, $w_1 = \frac{3(b-a)}{8} = w_2$ e

$$\int_a^b f(x)dx \approx \frac{b-a}{8}[f(a) + 3f(\frac{2a+b}{3}) + 3f(\frac{a+2b}{3}) + f(b)]$$

conhecida por *regra de Newton dos três oitavos*.

8.2.2 Erros de truncatura

Para cada uma das regras de Newton-Cotes, é possível deduzir a correspondente fórmula do erro de truncatura. Aplicando o Teorema 6.2 do Capítulo 6, que estabelece

$$f(x) - p_n(x) = \pi_n(x)\frac{f^{(n+1)}(\xi(x))}{(n+1)!}$$

para o erro da interpolação polinomial, $p_n(x)$, em relação à função $f(x)$, tem-se que o erro de truncatura da integração, de um modo geral, é definido por

$$\int_a^b f(x)dx = \int_a^b p_n(x)dx + E(x)$$

[4] A área do trapézio de bases iguais a $f(b)$ e $f(a)$ e altura igual a $b-a$ é usada para aproximar a área definida por $\int_a^b f(x)dx$.

[5] Esta fórmula para aproximar um integral definido, baseada em três pontos igualmente espaçados, foi desenvolvida, em 1612, por J. Kepler, na tentativa de calcular o volume de um barril de vinho. Foi redescoberta na sua forma geométrica por Cavalier em 1639 e, mais tarde, por Gregory (1668) e pelo próprio Simpson em 1743. O conceito de integral só foi desenvolvido nos finais do século dezassete. T. Simpson viveu entre 1710 e 1761 e o seu nome é conhecido pela fórmula de integração; no entanto, os trabalhos que desenvolveu noutros domínios, como a geometria, trigonometria, astronomia e teoria das probabilidades, tiveram consequências mais importantes. Foi ele que inventou os nomes para as funções trigonométricas: seno, coseno, tangente e cotangente (Goldstine (1977) e Hämmerlin e Hoffmann (1991)).

com

$$E(x) = \int_a^b \pi_n(x) \frac{f^{(n+1)}(\xi(x))}{(n+1)!}\, dx, \tag{8.2}$$

$\pi_n(x) = (x - x_0)(x - x_1)\ldots(x - x_n)$ e $a \leq x_0 < \ldots < x_n \leq b$.

Considere-se o Teorema do valor médio do cálculo integral:

Teorema 8.1 : Se as duas funções $f(x)$ e $g(x)$ são contínuas e se, além disso, g não muda de sinal no intervalo $[a, b]$, então existe um ponto $\xi \in [a, b]$ tal que

$$\int_a^b f(x)g(x)dx = f(\xi) \int_a^b g(x)dx\,.$$

Aplicando este teorema a (8.2) chega-se à seguinte expressão para o erro de integração

$$E(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \int_a^b \pi_n(x)\, dx \quad \text{com } \xi \in [a, b]\,. \tag{8.3}$$

Assim, em relação à *regra do rectângulo*, como $n = 0$ e $x_0 = a$, tem-se

$$E_R = \int_a^b (x - a) f'(\xi(x))dx = f'(\xi) \int_a^b (x - a)dx = \frac{(b-a)^2}{2} f'(\xi)\,, \quad \xi \in [a, b]$$

pois $(x - a)$ não muda de sinal em $[a, b]$.

Para a regra do ponto médio, $n = 0$, $x_0 = \frac{a+b}{2}$,

$$E_M = \int_a^b (x - \frac{a+b}{2}) f'(\xi(x))dx$$

e como $\pi_0(x) = (x - \frac{a+b}{2})$ muda de sinal em $[a, b]$ tem-se $\int_a^b \pi_0(x)dx = 0$, donde se conclui que não é possível aplicar o Teorema do valor médio do cálculo integral. Voltaremos a este caso mais adiante.

Para a *regra do trapézio*, tem-se

$$E_T = \frac{1}{2} \int_a^b (x - a)(x - b) f''(\xi(x))\, dx = \frac{1}{2} f''(\xi) \int_a^b (x - a)(x - b)\, dx = -\frac{(b-a)^3}{12} f''(\xi)$$

e o Teorema 8.1. pôde ser aplicado.

O mesmo já não acontece com a fórmula de Simpson. O erro, de acordo com (8.2), é dado por

$$E_S = \int_a^b (x - a)(x - \frac{a+b}{2})(x - b) \frac{f'''(\xi(x))}{3!} dx$$

mas, como $(x - a)(x - \frac{a+b}{2})(x - b)$ muda de sinal no ponto $\frac{a+b}{2}$ de $[a, b]$, o Teorema do valor médio não pode ser aplicado. Esta situação ocorrerá sempre que um *número impar de pontos igualmente distanciados* é usado na interpolação.

Falta deduzir a fórmula do erro para a *regra dos três oitavos*. Tem-se

$$E_{\frac{3}{8}} = \int_a^b \pi_3(x) \frac{f^{(iv)}(\xi(x))}{4!}\, dx = -\frac{(b-a)^5}{6480} f^{(iv)}(\xi)$$

uma vez que $\int_a^b \pi_3(x)\, dx = -\frac{(b-a)^5}{270}$ com $\pi_3(x) = (x-a)(x-\frac{2a+b}{3})(x-\frac{a+2b}{3})(x-b)$.

Voltando às fórmulas dos erros para as regras do ponto médio e de Simpson, utilizando um resultado que é possível deduzir (Isaacson e Keller (1966)) a partir da fórmula geral do resto (8.2) e introduzindo a função

$$\Pi_n(x) = \int_a^x \pi_n(z)\, dz,\ n = 1, 2, \ldots$$

tem-se

Lema 8.2 : Para n par

$$\Pi_n(a) = \Pi_n(b) = 0$$

e

$$\Pi_n(x) > 0,\ \textit{para}\ a < x < b.$$

Suponha, ainda, que a função integranda tem $n+2$ derivadas contínuas. Integrando por partes e usando o Lema 8.2, o erro

$$E(x) = \int_a^b \frac{d\Pi_n(x)}{dx} \frac{f^{(n+1)}(\xi(x))}{(n+1)!}\, dx,$$

reduz-se a

$$\begin{aligned} E(x) &= \Pi_n(x) \frac{f^{(n+1)}(\xi(x))}{(n+1)!} \,|_a^b - \int_a^b \Pi_n(x) \frac{d}{dx} \frac{f^{(n+1)}(\xi(x))}{(n+1)!}\, dx \\ &= -\int_a^b \Pi_n(x) \frac{f^{(n+2)}(\xi(x))}{(n+2)!}\, dx. \end{aligned}$$

Em virtude do estabelecido no Lema 8.2. é possível aplicar o Teorema do valor médio a $E(x)$, e

$$E(x) = -\frac{f^{(n+2)}(\xi)}{(n+2)!} \int_a^b \Pi_n(x)\, dx = \frac{f^{(n+2)}(\xi)}{(n+2)!} \int_a^b x\, \pi_n(x)\, dx, \tag{8.4}$$

em virtude de

$$\int_a^b \Pi_n(x)\, dx = x\Pi_n(x) \,|_a^b - \int_a^b x \frac{d}{dx} \Pi_n(x)\, dx = -\int_a^b x\, \pi_n(x)\, dx.$$

Assim, aplicando o resultado estabelecido em (8.4), obtém-se para a *regra do ponto médio*,

$$E_M = \frac{f''(\xi)}{2}\int_a^b x(x-\frac{a+b}{2})\,dx = \frac{(b-a)^3}{24}f''(\xi).$$

Para a *regra de Simpson* e novamente usando a expressão (8.4), chega-se a

$$E_S = \frac{f^{(iv)}(\xi)}{4!}\int_a^b x(x-a)(x-\frac{a+b}{2})(x-b)\,dx = -\frac{(b-a)^5}{2880}f^{(iv)}(\xi)\,.$$

Fica-se tentado a acreditar que a aproximação a um integral vai melhorando à medida que se aproxima a função integranda por polinómios de grau cada vez maior. No entanto, isto não é necessariamente verdadeiro. De facto, quanto maior é n pior é a interpolação nos pontos intermédios devido à natureza oscilante de $p_n(x)$.

Com a integração, e de um modo geral, os erros devidos às oscilações de $p_n(x)$ tendem a cancelar uns com os outros, embora, para funções $f(x)$ que variam muito rapidamente em certas regiões, isto não se verifica. Assim, as fórmulas de integração baseadas em polinómios de grau elevado só devem ser usadas quando $f(x)$ não varia muito no intervalo $[a,b]$.

Para ultrapassar facilmente este inconveniente pode-se interpolar a função $f(x)$ usando polinómios segmentados de grau baixo, integrando-os posteriormente em cada segmento. Esta abordagem é equivalente à divisão do intervalo $[a,b]$ em subintervalos, com extremos definidos pelos pontos $a = x_0 < x_1 < \ldots < x_{n-1} < x_n = b$, à utilização de uma regra de integração, de grau baixo, em cada um dos subintervalos $[x_i, x_{i+1}]$, $0 \leq i \leq n-1$ e à posterior soma de todas essas contribuições. As fórmulas de integração resultantes são conhecidas por *fórmulas compostas*.

8.2.3 Fórmulas compostas

Três das fórmulas compostas mais conhecidas são as que resultam da aplicação das regras do trapézio, de Simpson ou dos três oitavos em cada um dos subintervalos de $[a,b]$.

A *fórmula composta do trapézio* resulta da aplicação da interpolação segmentada linear. Isto é, em cada subintervalo, a função $f(x)$ é aproximada por um polinómio de grau um.

Assim, a aplicação da regra do trapézio, no subintervalo $[x_i, x_{i+1}]$, dá

$$\int_{x_i}^{x_{i+1}} f(x)\,dx \approx \frac{(x_{i+1}-x_i)}{2}[f(x_i)+f(x_{i+1})] \approx \frac{h}{2}[f_i + f_{i+1}]$$

para $i = 0, \ldots, n-1$, supondo que o espaçamento entre os pontos é constante e igual a h. Usa-se, para simplificar a notação, f_i no lugar de $f(x_i)$. O valor do integral no intervalo

$[a,b]$ obtém-se adicionando as contribuições dos n subintervalos, o que dá

$$\int_a^b f(x)dx = \sum_{i=0}^{n-1}\int_{x_i}^{x_{i+1}} f(x)\,dx \approx \frac{h}{2}\sum_{i=0}^{n-1}[f_i + f_{i+1}] \tag{8.5}$$

$$\approx \frac{h}{2}[f_0 + 2f_1 + 2f_2 + \cdots + 2f_{n-2} + 2f_{n-1} + f_n]. \tag{8.6}$$

Do mesmo modo, o erro de truncatura desta fórmula obtém-se somando os erros de truncatura de cada um dos n subintervalos,

$$\begin{aligned} E_T(h) &= -\sum_{i=0}^{n-1}\frac{(x_{i+1}-x_i)^3}{12}f''(\xi_i) = -\frac{h^2}{12}\frac{(b-a)}{n}\sum_{i=0}^{n-1} f''(\xi_i) \\ &= -\frac{h^2}{12}(b-a)f''(\eta)\,, \quad \eta \in [a,b] \end{aligned}$$

com $\xi_i \in [x_i, x_{i+1}]$ e $f''(\eta)$ é o valor médio das n derivadas $f''(\xi_i), i = 0, \ldots, n-1$.

A *fórmula composta de Simpson* resulta da aplicação da interpolação segmentada quadrática. Cada aplicação da regra de Simpson precisa de *dois subintervalos*. Se o intervalo $[a,b]$ for dividido em n (número par) subintervalos e $h = \frac{b-a}{n}$ é a amplitude constante, considere-se $m = \frac{n}{2}$.

Nos subintervalos $[x_{i-1}, x_{i+1}]$, a aplicação da regra de Simpson resulta em

$$\int_{x_{i-1}}^{x_{i+1}} f(x)\,dx \approx \frac{(x_{i+1}-x_{i-1})}{6}[f(x_{i-1}) + 4f(x_i) + f(x_{i+1})] \approx \frac{h}{3}[f_{i-1} + 4f_i + f_{i+1}]$$

para $i = 1(2\,\mathrm{em}\,2)n-1$. O integral em todo o intervalo $[a,b]$ é dado pela soma dos m conjuntos de subintervalos

$$\int_a^b f(x)dx = \sum_m \int_{x_{i-1}}^{x_{i+1}} f(x)\,dx \approx \frac{h}{3}\sum_m [f_{i-1} + 4f_i + f_{i+1}] \tag{8.7}$$

$$\approx \frac{h}{3}[f_0 + 4f_1 + 2f_2 + 4f_3 + \cdots + 4f_{n-3} + 2f_{n-2} + 4f_{n-1} + f_n] \tag{8.8}$$

em que $\sum_m$ representa o somatório dos m termos relativos aos m pares de subintervalos e equivale a fazer $i = 1(2\,\mathrm{em}\,2)n-1$.

A fórmula para o erro de truncatura é a seguinte

$$\begin{aligned} E_S(h) &= -\sum_m \frac{(x_{i+1}-x_{i-1})^5}{2880}f^{(iv)}(\xi_i) = -\frac{(2h)^4}{2880}\frac{(b-a)}{m}\sum_m f^{(iv)}(\xi_i) \\ &= -\frac{h^4}{180}(b-a)f^{(iv)}(\eta)\,, \quad \eta \in [a,b] \end{aligned}$$

com $\xi_i \in [x_{i-1}, x_{i+1}]$, $2h = \frac{b-a}{m}$ e $f^{(iv)}(\eta)$ é o valor médio das m derivadas $f^{(iv)}(\xi_i)$.

A *fórmula composta de Newton dos* $\frac{3}{8}$ resulta da aplicação da regra dos $\frac{3}{8}$ a cada conjunto de três subintervalos e equivale a uma interpolação cúbica em cada segmento de três subintervalos. Se o intervalo $[a,b]$ for dividido em n (número múltiplo de 3) subintervalos e $h=\frac{b-a}{n}$, considere-se $r=\frac{n}{3}$.

No subintervalo $[x_{i-1},x_{i+2}]$, tem-se

$$\begin{aligned}\int_{x_{i-1}}^{x_{i+2}} f(x)\,dx &\approx \frac{(x_{i+2}-x_{i-1})}{8}[f(x_{i-1})+3f(x_i)+3f(x_{i+1})+f(x_{i+2})]\\ &\approx \frac{3h}{8}[f_{i-1}+3f_i+3f_{i+1}+f_{i+2}]\end{aligned}$$

para $i=1(3\,\text{em}\,3)n-2$. O valor do integral em $[a,b]$ é dado pelo somatório relativo a todos os r conjuntos de três subintervalos onde se aplicou a regra dos três oitavos,

$$\begin{aligned}\int_a^b f(x)dx &= \sum_r \int_{x_{i-1}}^{x_{i+2}} f(x)\,dx \approx \frac{3h}{8}\sum_r [f_{i-1}+3f_i+3f_{i+1}+f_{i+2}] \qquad (8.9)\\ &\approx \frac{3h}{8}[f_0+3f_1+3f_2+2f_3+\cdots+2f_{n-3}+3f_{n-2}+3f_{n-1}+f_n] \qquad (8.10)\end{aligned}$$

em que $\sum_r$ representa o somatório dos r termos relativos aos r trios de subintervalos e equivale a fazer $i=1(3\,\text{em}\,3)n-2$.

Para a fórmula do erro de truncatura, tem-se

$$\begin{aligned}E_{\frac{3}{8}}(h) &= -\sum_r \frac{(x_{i+2}-x_{i-1})^5}{6480} f^{(iv)}(\xi_i) = -\frac{(3h)^4}{6480}\frac{(b-a)}{r}\sum_r f^{(iv)}(\xi_i)\\ &= -\frac{h^4}{80}(b-a)f^{(iv)}(\eta)\,, \quad \eta\in[a,b]\end{aligned}$$

em que cada ξ_i pertence a $[x_{i-1},x_{i+2}]$, $3h=\frac{b-a}{r}$ e $f^{(iv)}(\eta)$ é o valor médio das r derivadas $f^{(iv)}(\xi_i)$.

8.2.4 Extensão a intervalos com amplitudes variadas

Quando o espaçamento entre os pontos x_i, da tabela de valores de $f(x)$, não é constante, as fórmulas de Newton-Cotes e as correspondentes fórmulas composta podem ainda assim ser utilizadas para calcular o integral. É possível *combinar* num mesmo integral várias fórmulas de integração. O cálculo de

$$I=\int_a^b f(x)\,dx$$

pode ser feito a partir de

$$\int_{x_0}^{x_i} f(x)\,dx+\int_{x_i}^{x_j} f(x)\,dx+\cdots+\int_{x_k}^{x_n} f(x)\,dx \qquad (8.11)$$

se os pontos x_r pertencem ao intervalo $[a, b]$ e

$$a = x_0 < x_1 < x_2 < \ldots < x_i < \ldots < x_j < \ldots < x_k < \ldots < x_n = b.$$

Pode-se aplicar uma regra ou fórmula composta de integração em cada integral de (8.11), desde que em cada conjunto de subintervalos $[x_0, x_i]$, $[x_i, x_j]$, $\ldots$, $[x_k, x_n]$ o espaçamento entre os pontos lá existentes seja constante. Dependendo do número de pontos dentro de cada conjunto e da aproximação desejada (erro de truncatura), assim se aplica, em cada conjunto de subintervalos a fórmula mais adequada.

As normas que devem ser seguidas para a aplicação das regras ou fórmulas compostas de integração, a cada conjunto de pontos igualmente distanciados, são:

1. aplica-se a regra do trapézio a cada conjunto definido por um único segmento, $[x_i, x_{i+1}]$,
2. aplica-se a fórmula composta do trapézio a cada conjunto formado por um número impar (diferente de múltiplo de três) de subintervalos de amplitudes iguais,
3. aplica-se a regra de Simpson a cada conjunto definido por dois subintervalos iguais,
4. aplica-se a fórmula composta de Simpson a cada conjunto formado por um número par de subintervalos de amplitudes iguais,
5. aplica-se a regra dos três oitavos a cada conjunto formado por três subintervalos iguais,
6. aplica-se a fórmula composta dos três oitavos a cada conjunto que seja formado por um múltiplo de três subintervalos, todos com a mesma amplitude.

Estas normas são relativas apenas às três últimas regras deduzidas em 8.2.1 e às fórmulas compostas apresentadas em 8.2.3.

O algoritmo 8.1 calcula numericamente o integral de $f(x)$, se esta for dada por uma tabela de valores, pesquisando ao longo da tabela a possibilidade de aplicação das fórmulas de integração, de acordo com as normas descritas. Isto é, se aparecer um número múltiplo de dois subintervalos iguais, aplica a fórmula de Simpson; se aparecer um número múltiplo de três subintervalos de amplitudes iguais, aplica a fórmula dos três oitavos; e para os outros casos aplica a fórmula do trapézio.

Algoritmo 8.1 :

1. `ler` n `e para` $i = 0, \ldots, n$ `ler` x_i `e` f_i `( ou calcular` $f_i = f(x_i)$ `)`
2. `fazer` $k = 0$ `,` $y_{01} = f_0$ `e` $y_{11} = f_1$
3. `calcular` $h_1 = x_1 - x_0$ `, fazer` $nint_1 = 1$ `e` $j = 1$
4. `para` $i = 1, \ldots, n-1$

   4.1. `fazer` $k = k + 1$ `,` $y_{kj} = f_i$ `e calcular` $dif = x_{i+1} - x_i$

4.2. se $dif = h_j$ então fazer $nint_j = nint_j + 1$

4.3. senão fazer $j = j + 1$, $k = 0$, $h_j = dif$, $y_{kj} = f_i$ e $nint_j = 1$

5. fazer $y_{k+1\,j} = f_n$, $nconj = j$ e $integral = 0$

6. para $k = 1, \ldots, nconj$

6.1. se $nint_k = mltiplo\,de\,2$ então (regra de Simpson) fazer $S = 0$

6.1.1. para $i = 0, \ldots, nint_k$

6.1.1.1. se $i = 0$ ou $i = nint_k$ então calcular $S = S + y_{ik}$

6.1.1.2. senão

6.1.1.2.1. se $i = mltiplo\,de\,2$ então calcular $S = S + 2y_{ik}$

6.1.1.2.2. senão calcular $S = S + 4y_{ik}$

6.1.2. calcular $termo = \frac{h_k}{3} S$

6.2. senão, se $nint_k = mltiplo\,de\,3$ então (regra dos 3/8's) fazer $O = 0$

6.2.1. para $i = 0, \ldots, nint_k$

6.2.1.1. se $i = 0$ ou $i = nint_k$ então calcular $O = O + y_{ik}$

6.2.1.2. senão

6.2.1.2.1. se $i = mltiplo\,de\,3$ então calcular $O = O + 2y_{ik}$

6.2.1.2.2. senão calcular $O = O + 3y_{ik}$

6.2.2. calcular $termo = \frac{3h_k}{8} O$

6.2.3. senão (regra do trapézio) fazer $T = 0$

6.2.3.1. para $i = 0, \ldots, nint_k$

6.2.3.1.1. se $i = 0$ ou $i = nint_k$ então calcular $T = T + y_{ik}$

6.2.3.1.2. senão calcular $T = T + 2y_{ik}$

6.2.3.2. calcular $termo = \frac{h_k}{2} T$

6.3. calcular $integral = integral + termo$

7. terminar com $I \leftarrow integral$.

8.2.5 Implementação. Rotina QUADRA

A rotina QUADRA, que implementa o algoritmo 8.1, foi usada para resolver os seguintes exemplos:

Exemplo 8.1 No cálculo do integral

$$I = \int_0^1 \frac{4}{1+x^2}\, dx$$

cujo valor exacto é π, obtém-se o valor 3.141569, se o espaçamento entre pontos for constante e igual a $h = 0.25$. Como o número de subintervalos é 4, o algoritmo 8.1 selecciona a fórmula de Simpson.

Se for escolhido um espaçamento constante e igual a $h = 0.0625$, o número de subintervalos é igual a 16 e mais uma vez é usada a fórmula de Simpson. O valor obtido para o integral é 3.141593.

Se forem usados os pontos: $0.0, 0.1, 0.2, 0.3, 0.5, 0.7, 0.8, 0.9, 1.0$, tem-se um número par de subintervalos, mas o espaçamento não é constante. Por isso, vai ser usada uma combinação de fórmulas. De 0.0 a 0.3 existem três subintervalos com espaçamento constante e igual a $h_1 = 0.1$. De 0.3 a 0.7 tem-se um par de subintervalos com $h_2 = 0.2$. De 0.7 a 1.0 o espaçamento é de $h_3 = 0.1$ e define um conjunto de três subintervalos. Assim, são utilizadas as seguintes fórmulas: três oitavos, Simpson e depois três oitavos. O valor calculado final, soma destas três contribuições, é 3.141484.

Exemplo 8.2 Para calcular

$$I = \int_{-0.25}^{0.25} e^x \, dx$$

pode usar-se apenas os dois pontos extremos do intervalo. Como $n = 1$ subintervalo, a fórmula do trapézio dá 0.515707. Se for escolhido mais o ponto 0.00, passa-se a ter dois subintervalos e a fórmula de Simpson calcula $I \approx 0.505236$. Considerando ainda outro caso, em que $h = 0.125$, constante, os pontos passam a ser ao todo cinco: $-0.25, -0.125, 0.00, 0.125, 0.25$. Como $n = 4$ o algoritmo 8.1 selecciona a fórmula de Simpson e dá $I \approx 0.505225$.

8.3 Integração de Romberg

8.3.1 Introdução teórica

A *técnica de Romberg* pode ser usada com as fórmulas composta de integração com o objectivo de melhorar a aproximação calculada, uma vez que a sua aplicação diminui o erro de truncatura. Preferencialmente a técnica de Romberg tem sido implementada juntamente com a fórmula do trapézio. O integral, $\int_a^b f(x)\,dx$, é aproximado pelo valor

$$T(h) = \frac{h}{2}[f_0 + 2f_1 + 2f_2 + \cdots + 2f_{n-2} + 2f_{n-1} + f_n]$$

quando é calculado pela fórmula composta do trapézio. O erro de truncatura desta aproximação é função de h e é dado por

$$E_T(h) = -\frac{(b-a)}{12} h^2 f''(\eta), \quad \eta \in [a, b]$$

e que pode ser expandido na forma

$$E_T(h) = Ah^2 + Bh^4 + \cdots + Gh^{2m-2} + \mathcal{O}(h^{2m}), \tag{8.12}$$

dominado pelo termo em função de h^2, para pequenos valores do espaçamento h. O coeficiente A depende da segunda derivada de f, o coeficiente B da expansão (8.12) depende da derivada de ordem quatro de f e assim sucessivamente.

Calculando o valor de $T(h)$ para vários valores de h, por exemplo, h_1 e h_2, os erros destas aproximações podem ser colocados na forma

$$I - T(h_1) = Ah_1^2 + Bh_1^4 + Ch_1^6 + \cdots$$

$$I - T(h_2) = Ah_2^2 + Bh_2^4 + Ch_2^6 + \cdots$$

e, é possível, a partir destes valores, calcular outra aproximação ao integral, com menor erro de truncatura do que qualquer destes. Para se conseguir um erro menor, este terá de ser dominado pelo segundo termo, que é função, de um modo geral, de h^4. Isto significa que o primeiro termo da expressão do erro deve anular-se. Para fazer desaparecer o termo do erro em função de h^2, basta adicionar $h_2^2(I - T(h_1))$ a $-h_1^2(I - T(h_2))$. Assim

$$h_2^2(I - T(h_1)) = Ah_1^2h_2^2 + Bh_1^4h_2^2 + Ch_1^6h_2^2 + \cdots,$$

$$-h_1^2(I - T(h_2)) = -Ah_2^2h_1^2 - Bh_2^4h_1^2r - Ch_2^6h_1^2 + \cdots$$

e

$$I(h_2^2 - h_1^2) - (T(h_1)h_2^2 - T(h_2)h_1^2) = -B(-h_1^4h_2^2 + h_2^4h_1^2) - C(-h_1^6h_2^2 + h_2^6h_1^2) + \cdots$$

ou, dividindo por $h_2^2 - h_1^2$

$$I - T(h_1, h_2) = -Bh_1^2h_2^2 - Ch_1^2h_2^2(h_2^2 + h_1^2) - \cdots \tag{8.13}$$

em que $T(h_1, h_2)$ é dado por

$$\begin{aligned} T(h_1, h_2) &= \frac{T(h_1)h_2^2 - T(h_2)h_1^2}{h_2^2 - h_1^2} \\ &= \frac{T(h_1)h_2^2 - T(h_2)h_1^2 + T(h_2)h_2^2 - T(h_2)h_2^2}{h_2^2 - h_1^2} \\ &= \frac{T(h_2)(h_2^2 - h_1^2) + h_2^2(T(h_1) - T(h_2))}{h_2^2 - h_1^2} \\ &= T(h_2) + \frac{h_2^2}{h_2^2 - h_1^2}(T(h_1) - T(h_2)). \end{aligned}$$

Esta expressão final da nova aproximação é conseguida como uma combinação linear de duas aproximações, já calculadas pela fórmula do trapézio,

$$T(h_1, h_2) = T(h_2) + \frac{h_2^2}{h_1^2 - h_2^2}(T(h_2) - T(h_1)) \tag{8.14}$$

e, tem erro de truncatura dominado pelo primeiro termo de (8.13), que é função de $h_1^2h_2^2$ e que em termos gerais se considera da ordem de h^4.

Para melhorar ainda mais a aproximação ao integral, isto é, calcular outra aproximação, cujo erro de truncatura seja ainda menor do que os anteriores, por isso dominado pelo termo que é função, genericamente, de h^6, implementa-se novamente a técnica de Romberg. Usando a expansão do tipo (8.13) vai-se anular o primeiro termo do lado direito com a ajuda de

$$I - T(h_1, h_2) = -Bh_1^2h_2^2 - Ch_1^2h_2^2(h_1^2 + h_2^2) - \cdots$$

e

$$I - T(h_2, h_3) = -Bh_2^2h_3^2 - Ch_2^2h_3^2(h_2^2 + h_3^2) - \cdots \tag{8.15}$$

A expansão (8.15) corresponde à implementação da técnica de Romberg a $T(h_2)$, já conhecido, e a $T(h_3)$, novo valor obtido pela fórmula composta do trapézio para um terceiro valor de h, h_3. Assim, multiplicando (8.13) por h_3^2 e (8.15) por $-h_1^2$ e somando as equações resultantes, fica-se com

$$\begin{aligned} I(h_3^2 - h_1^2) &- (T(h_1, h_2)h_3^2 - T(h_2, h_3)h_1^2) = \\ &= -C[h_1^2h_2^2h_3^2(h_1^2 + h_2^2) - h_2^2h_3^2h_1^2(h_2^2 + h_3^2)] - \cdots \end{aligned}$$

que dividindo por $h_3^2 - h_1^2$ reduz-se a

$$I - T(h_1, h_2, h_3) = Ch_1^2h_2^2h_3^2 - \cdots$$

com $T(h_1, h_2, h_3)$ definido por

$$T(h_1, h_2, h_3) = T(h_2, h_3) + \frac{h_3^2}{h_1^2 - h_3^2}[T(h_2, h_3) - T(h_1, h_2)]. \tag{8.16}$$

Face à expressão do erro de truncatura desta aproximação, conclui-se que ele é da ordem de h^6. Tendo usado os seguintes valores de h, h_1, h_2 e h_3 o erro é dado por um múltiplo de $h_1^2h_2^2h_3^2$.

Este processo pode continuar e a aproximação seguinte terá um erro de truncatura da ordem de h^8. O seu valor consegue-se a partir de quatro valores obtidos pela fórmula composta do trapézio para os seguintes valores de h: h_1, h_2, h_3 e h_4. Assim, tem-se

$$T(h_1, h_2, h_3, h_4) = T(h_2, h_3, h_4) + \frac{h_4^2}{h_1^2 - h_4^2}[T(h_2, h_3, h_4) - T(h_1, h_2, h_3)] \tag{8.17}$$

A tabela que corresponde às aproximações assim calculadas é a seguinte:

$T(h_1)$				
	$T(h_1, h_2)$			
$T(h_2)$		$T(h_1, h_2, h_3)$		
	$T(h_2, h_3)$		$T(h_1, h_2, h_3, h_4)$	$\ldots$
$T(h_3)$		$T(h_2, h_3, h_4)$		
	$T(h_3, h_4)$			
$T(h_4)$				
$\ldots$				

Se os diferentes valores de h forem escolhidos por forma a verificarem-se as relações:

$$h_2 = \frac{1}{2}h_1, \quad h_3 = \frac{1}{2}h_2 = \frac{1}{4}h_1, \quad h_4 = \frac{1}{2}h_3 = \frac{1}{8}h_1, \ \ldots$$

os cálculos simplificam-se, uma vez que as expressões

$$\frac{h_2^2}{h_1^2 - h_2^2},\ \frac{h_3^2}{h_1^2 - h_3^2},\ \frac{h_4^2}{h_1^2 - h_4^2}, \ldots$$

das diversas fases da técnica de Romberg, reduzem-se, respectivamente a

$$\frac{1}{(2^{2n} - 1)},\ \text{para}\ n = 1, 2, 3, \ldots \tag{8.18}$$

não dependendo dos valores de h utilizados nos cálculos. Assim, para se obter uma aproximação da ordem de h^4, o coeficiente é ($n = 1$) igual $\frac{1}{3}$, para se obter uma aproximação da ordem de h^6, o coeficiente é de $\frac{1}{15}$ ($n = 2$), para uma aproximação da ordem de h^8, a expressão (8.18) dá, para $n = 3$, o valor $\frac{1}{63}$ e assim por adiante.

O algoritmo que implementa a técnica de Romberg é o algoritmo 8.2.

Algoritmo 8.2 :

1. ler $np = n + 1$ e $h_1 = x_n - x_0$
2. se $np - 1 \neq potncia\,de\,2$ então recomeçar, ir para o passo 1
3. fazer $n = np - 1$ e para $i = 0, \ldots, n$ ler x_i e f_i (ou calcular $f_i = f(x_i)$)
4. fazer $k = 0$, $l = 0$
5. fazer $den = 2^l$, $k = k + 1$, $j = \frac{n}{den}$
6. se $j\,no\,nmero\,inteiro$ então termina o processo, ir para o passo 11
7. para $i = 0, \ldots, den$ fazer $y_i = f_{i \times j}$, $h_k = \frac{h_{k-l}}{den}$ e $T = 0$
8. para $i = 0, \ldots, den$

   8.1. se $i = 0$ ou $i = den$ então calcular $T = T + y_i$

   8.2. senão calcular $T = T + 2y_i$

9. calcular $R_{k0} = \frac{h_k}{2} T$
10. se $den \leq n$ então fazer $l = l + 1$ e ir para o passo 5
11. construir a tabela: para $l = k - 2, \ldots, 1$ fazer $col = k - l - 1$
    para $i = 1, \ldots, l$ calcular $R_{i\,col} = R_{i+1\,col-1} + \frac{1}{(2^{2col}-1)}[R_{i+1\,col-1} - R_{i\,col-1}]$
12. calcular $est = \prod_{i=1}^{k-1} h_i^2$
13. terminar com $I \leftarrow R_{1\,k-2}$, com erro da ordem de est.

8.3.2 Implementação. Rotina ROMBER

A rotina ROMBER implementa o algoritmo 8.2. com os seguintes resultados:

Exemplo 8.3 Dada a tabela de valores da função $f(x)$

x_i	0.0	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0
f_i	-4271	-2522	-499	1795	4358	7187	10279	13633	17247

com oito$= 2^3$ subintervalos, obtém-se a tabela de Romberg

25952.0			
	20272.000000		
21692.0		20270.577778	
	20270.666667		20270.668078
20626.0		20270.666667	
	20270.666667		
20359.5			

em que os valores da primeira coluna correspondem respectivamente a $T(4.0), T(2.0), T(1.0)$ e $T(0.5)$, valores obtidos pela fórmula composta do trapézio para o integral $\int_0^4 f(x)\,dx$. O resultado 20270.668078 tem erro de truncatura da ordem de h^8. Como os valores de h utilizados são: 4.0, 2.0, 1.0 e 0.5, este valor tem um erro de truncatura que é um múltiplo de 16. Da tabela, e para este caso em que alguns valores de h são considerados grandes (> 1), possivelmente o melhor valor, com menor erro de truncatura, é 20270.666667 com um erro que é múltiplo de $0.5^2 \times 1^2$. Nestes comentários não se teve em conta os valores das derivadas, que para este caso sofrem grandes variações no intervalo.

Exemplo 8.4 A tabela de nove valores da função f, para valores de x espaçados de $h = 0.1$ é a seguinte

x_i	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
f_i	1.00000	0.99833	0.99334	0.98507	0.97355	0.95885	0.94107	0.92031	0.89670

As aproximações ao integral

$$\int_{0.0}^{0.8} f(x)\,dx$$

pela fórmula composta do trapézio são $0.758680, 0.768760, 0.771262$ e 0.771887, respectivamente para os seguintes valores de h: $0.8, 0.4, 0.2$ e 0.1. A tabela correspondente à implementação da técnica de Romberg é

0.758680			
	0.772120		
0.768760		0.772094	
	0.772096		0.772095
0.771262		0.772095	
	0.772095		
0.771887			

e, neste caso, a melhor aproximação ao integral é 0.772095, com erro de truncatura que é múltiplo de 0.000041.

8.4 Fórmulas Gaussianas

8.4.1 Introdução teórica

Até agora foram consideradas fórmulas de quadratura numérica baseadas num conjunto predeterminado de pontos. As fórmulas compostas de integração aparecem da divisão do intervalo $[a, b]$ em n subintervalos de amplitudes iguais. Quando a função $f(x)$ pode ser calculada para qualquer ponto do intervalo, torna-se mais económico usar subintervalos com amplitudes diferentes. Nesta secção, as fórmulas Gaussianas que vão ser estudadas para aproximar o integral, têm a forma

$$\int_a^b f(x)\,dx \approx A_0 f(x_0) + A_1 f(x_1) + \cdots + A_n f(x_n)$$

em que tanto os coeficientes A_i como os pontos x_i, $i = 0, \ldots, n$ são parâmetros a determinar. O objectivo é calcular estes parâmetros por forma a definir uma fórmula que dê resultados com a maior precisão possível, isto é, com o menor erro de truncatura.

Considerem-se os seguintes teoremas:

Teorema 8.3. (Quadratura Gaussiana) Seja $w(x)$ uma função peso positiva no intervalo $[a, b]$. Se os pontos $x_0, x_1, \ldots, x_n$ forem escolhidos como os zeros do polinómio $P_{n+1}(x)$ de grau $n + 1$, que pertence à família de polinómios ortogonais associados a $w(x)$, então a fórmula Gaussiana

$$\int_a^b w(x) f(x)\,dx \approx A_0 f(x_0) + A_1 f(x_1) + \ldots + A_n f(x_n)\,, \tag{8.19}$$

dá resultados exactos para todos os polinómios de grau menor ou igual a $2n + 1$. Os coeficientes são calculados através de

$$A_i = \int_a^b w(x) L_i(x)\,dx \tag{8.20}$$

sendo os $L_i(x)$ os polinómios de Lagrange (Capítulo 6, em 6.3.5.).

Teorema 8.4. Os coeficientes A_i nas fórmulas de quadratura Gaussiana são positivos.

Teorema 8.5. O erro de truncatura na quadratura Gaussiana é dado pela fórmula

$$\frac{f^{(2n+2)}(\xi)}{(2n+2)!} \int_a^b w(x)[\pi_n(x)]^2\,dx = c_n f^{(2n+2)}(\xi)\,,\ \ \xi \in (a, b)$$

em que $\pi_n(x) = \prod_{i=0}^{n}(x - x_i)$ e a constante c_n pode ser determinada aplicando a fórmula a algum polinómio de grau $2n + 2$.

Se o problema consiste em aproximar o integral da forma

$$\int_{-1}^{+1} w(x) f(x)\,dx \quad \text{com } \ w(x) = 1 \ \text{ para } \ x \in [-1, +1]$$

então a combinação (8.19) define a *fórmula de Gauss-Legendre*. Os $n+1$ argumentos, que são os zeros do polinómio ortogonal de Legendre de grau $n+1$, e os coeficientes, calculados de acordo com (8.20), estão representados na tabela 8.1, para vários valores de n,

n	x_i	A_i
1	$\pm$ 0.5773503	1
2	0	0.8888889
	$\pm$ 0.7745967	0.5555556
3	$\pm$ 0.3399810	0.6521452
	$\pm$ 0.8611363	0.3478548
4	0	0.5688889
	$\pm$ 0.5384693	0.4786287
	$\pm$ 0.9061798	0.2369269
5	$\pm$ 0.2386192	0.4679139
	$\pm$ 0.6612094	0.3607616
	$\pm$ 0.9324695	0.1713245

Tabela 8.1: Argumentos e coeficientes da fórmula de Gauss-Legendre

Quando se assume que o intervalo $[a,b]$ é $[-1,+1]$, não se está a particularizar demasiado. Qualquer intervalo $[a,b]$ pode ser transformado em $[-1,+1]$ através da relação

$$t = 2\frac{x-a}{b-a} - 1\,. \tag{8.21}$$

Nesta situação

$$\begin{aligned}\int_a^b f(x)\,dx &= \int_{-1}^{1} f(\frac{b-a}{2}t + \frac{b+a}{2})\ \frac{b-a}{2}\,dt \\ &= \int_{-1}^{1} g(t)\,dt\end{aligned}$$

em que $g(t)$ é a função de t que se obtém a partir de $f(x)$, quando se substitui x pela expressão em função de t, multiplicada por dx, com

$$x = \frac{b-a}{2}t + \frac{b+a}{2} \quad \text{e} \quad dx = \frac{b-a}{2}\,dt.$$

Em algumas aplicações torna-se vantajoso a fixação de um ponto do intervalo de integração $[-1,+1]$, nomeadamente um dos pontos limites do intervalo, ou mesmo os dois limites. A versão *fórmula de Gauss-Radau* surge quando se fixa um dos limites. Isto é, ou o ponto -1 é um dos argumentos, ou é o ponto $+1$. Se,

(i) $x_0 = -1$, então pretende-se determinar os argumentos $x_1, x_2, \ldots, x_n$ de tal forma que todos os polinómios de grau menor ou igual a $2n$ sejam integrados exactamente.

Eles são os zeros do polinómio ortogonal, de grau $n-1$, caracterizado pelo intervalo $[-1,+1]$ e pela função peso $w(x)=x+1$. Tem-se, assim,

$$\int_{-1}^{+1} f(x)\,dx \approx \frac{2}{(n+1)^2} f(-1) + \sum_{i=1}^{n} A_i f(x_i)$$

quando o ponto fixo é $x_0=-1$. Os restantes argumentos e correspondentes coeficientes encontram-se na tabela 8.2, nas duas colunas da esquerda, indexadas com x_i e A_i.

Do mesmo modo

$$\int_{-1}^{+1} f(x)\,dx \approx \frac{2}{(n+1)^2} f(+1) + \sum_{i=0}^{n-1} A_i f(x_i)$$

quando o ponto fixo é $x_n=+1$. Os argumentos desta versão são simétricos dos do caso anterior e os coeficientes que lhes correspondem são os mesmos.

n	x_i	A_i	x_i	A_i
1	-1	0.5		
	0.333333	1.5		
2	-1	0.222222	regra	
	-0.289898	1.024972	de	
	0.689898	0.752806	Simpson	
3	-1	0.125	$\pm$ 0.447214	0.833333
	-0.575319	0.657689	$\pm$ 1	0.166667
	0.181066	0.776387		
	0.822824	0.440924		
4	-1	0.08	0	0.711111
	-0.720480	0.446208	$\pm$ 0.654654	0.544444
	-0.167181	0.623653	$\pm$ 1	0.1
	0.446314	0.562712		
	0.885792	0.287427		

Tabela 8.2: Argumentos e coeficientes das fórmulas de Gauss-Radau e Gauss-Lobatto

(ii) $x_0=-1$ e $x_n=+1$, os restantes argumentos são determinados como os zeros de um polinómio ortogonal, de grau $n-2$, em relação à função peso $w(x)=1-x^2$. A fórmula resultante é simétrica e chama-se *fórmula de Gauss-Lobatto*,

$$\int_{-1}^{+1} f(x)\,dx \approx \frac{2}{(n+1)n}[f(-1)+f(+1)] + \sum_{i=1}^{n-1} A_i f(x_i)\,.$$

A tabela 8.2. apresenta os argumentos e coeficientes para vários valores de n, nas duas colunas da direita indexadas com x_i e A_i.

8.4.2 Integrais impróprios

Integral impróprio é aquele em que, ou a função integranda $f(x)$ tem uma singularidade no intervalo de integração $[a, b]$, ou um dos limites de integração, ou mesmo os dois, são infinitos.

Existem várias técnicas para ultrapassar estas dificuldades:

1. Remover a singularidade de $f(x)$. Se a singularidade se verifica no ponto $c \in [a, b]$, então o integral $\int_a^b f(x)dx$ pode ser calculado se

$$lim_{\epsilon \to 0}[\int_a^{c-\epsilon} f(x)\,dx + \int_{c+\epsilon}^b f(x)\,dx]$$

existir.

2. Utilizar a fórmula de Gauss-Legendre. Se a singularidade se verifica num dos extremos (ou em ambos) do intervalo, a utilização da fórmula de Gauss-Legendre pode ser adequada uma vez que ela não utiliza os extremos do intervalo como argumentos.

3. Truncar o intervalo. Se um dos extremos do intervalo é infinito, truncar o intervalo e usar um método para integrais definidos na parte finita, fornece bons resultados desde que o 'restante' tenha uma contribuição insignificante. Assim,

$$I = \int_0^\infty f(x)\,dx = \int_0^a f(x)\,dx + \int_a^\infty f(x)\,dx \approx \int_0^a f(x)\,dx$$

em que a deve ser escolhido por forma a que

$$|\int_a^\infty f(x)\,dx| < \epsilon(a)$$

com $\epsilon(a)$ uma quantidade positiva e próxima de zero. O valor adequado de a raramente é conhecido, no entanto, pode ser determinado usando o seguinte processo: escolhe-se um conjunto de valores que satisfazem $0 < a_1 < a_2 < \cdots < a_k < \cdots$ e calcula-se sucessivamente os integrais

$$\int_0^{a_1} f(x)dx, \int_{a_1}^{a_2} f(x)dx, \ldots, \int_{a_k}^{a_{k+1}} f(x)dx, \ldots$$

Quando for encontrado um integral cuja contribuição se torna insignificante, em relação ao total já calculado, pode-se concluir que o integral a partir desse valor até ao infinito é também insignificante. Se este processo não convergir, pode-se considerar que o integral não é definido.

4. Transformar o intervalo. Se o intervalo for $[0, \infty)$, uma transformação de variáveis do tipo $x = -ln(t)$ ou $x = \frac{t}{(1-t)}$ transforma o intervalo dado em $[0, 1]$. Por exemplo,

$$\int_0^\infty f(x)\,dx = -\int_1^0 f(-ln(t))\,\frac{dt}{t} = \int_0^1 f(-ln(t))\,\frac{dt}{t}\,.$$

A transformação $x = ln(\frac{1+t}{1-t})$ transforma o integral $(-\infty, +\infty)$ em $(-1, +1)$. As transformações têm de ser usadas com muito cuidado, ou o objectivo, de obter um integral mais simples e não apenas um integral finito, não será atingido.

5. Utilizar uma das seguintes fórmulas de Gauss:

   Gauss-Laguerre, Gauss-Hermite ou Gauss-Chebyshev.

   Se a função integranda $g(x)$ pode ser colocada na forma $w(x)\, f(x)$, em que $f(x)$ se comporta como um polinómio e $w(x)$ é uma função sempre positiva no intervalo de integração, a utilização de quadratura Gaussiana traz vantagens uma vez que evita cálculos com a função $w(x)$, que normalmente tem problemas de singularidade.

   Considerem-se estes três casos:

As fórmulas de quadratura para aproximar integrais impróprios da forma

$$\int_a^b g(x)\, dx = \int_a^b w(x)\, f(x)\, dx,$$

em que o intervalo não é limitado, ou a função $w(x)$ tem singularidades nos extremos do intervalo $[a, b]$, mais conhecidas são as de Gauss-Laguerre, Gauss-Hermite e Gauss-Chebyshev.

i) Para calcular

$$\int_0^\infty e^{-x} f(x)\, dx$$

os polinómios ortogonais definidos no intervalo $[0, \infty)$ em relação à função peso $w(x) = e^{-x}$ são os polinómios de Laguerre. A fórmula resultante de aproximação ao integral chama-se *Gauss-Laguerre* e

$$\int_0^\infty e^{-x} f(x)\, dx \approx A_0 f(x_0) + \ldots + A_n f(x_n)$$

em que os argumentos x_i são os zeros do polinómio de Laguerre de grau $n+1$. Os coeficientes são determinados pelo processo indicado no Teorema 8.3. Na tabela 8.3 apresentam-se os argumentos e coeficientes da fórmula de Gauss-Laguerre para quatro valores de n, nas duas colunas da esquerda indexadas com x_i e A_i.

ii) No cálculo de

$$\int_{-\infty}^{+\infty} e^{-x^2} f(x)\, dx$$

os polinómios ortogonais definidos no intervalo $(-\infty, +\infty)$ relativos a $w(x) = e^{-x^2}$ são os polinómios de Hermite. A *fórmula de Gauss-Hermite* aproxima o integral

$$\int_{-\infty}^{+\infty} e^{-x^2} f(x)\, dx \approx A_0 f(x_0) + \cdots + A_n f(x_n)$$

em que os argumentos, que são os zeros do polinómio de Hermite de grau $n+1$, e os coeficientes estão representados na tabela 8.3, nas duas colunas da direita.

n	x_i	A_i	x_i	A_i
0	1	1	0	1.772454
1	0.585786	0.853553	$\pm$ 0.707107	0.886227
	3.414214	0.146447		
2	0.415775	0.711093	0	1.181636
	2.294280	0.278518	$\pm$ 1.224745	0.295409
	6.289945	0.0103893		
3	0.322548	0.603154	$\pm$ 0.524648	0.804914
	1.745761	0.357419	$\pm$ 1.650680	0.0813128
	4.536620	0.0388879		
	9.395071	0.0005393		

Tabela 8.3: Argumentos e coeficientes das fórmulas de Gauss-Laguerre e Gauss-Hermite

iii) Os polinómios ortogonais de Chebyshev $T_n(x) = cos(n\, arccos(x))$ são usados no cálculo do integral

$$\int_{-1}^{+1} \frac{f(x)}{\sqrt{1-x^2}}\, dx.$$

Os argumentos desta fórmula de Gauss são dados por

$$x_i = cos(\frac{(2i+1)\pi}{2n+2}),\ 0 \leq i \leq n$$

e os coeficientes são todos iguais a $\frac{\pi}{(n+1)}$.

A escolha da melhor técnica depende fortemente das características do problema. Infelizmente, com integrais impróprios, não é possível aconselhar a utilização de uma determinada técnica que possa ser usada na maior parte dos casos. O algoritmo 8.3 calcula integrais pelas fórmulas de Gauss-Legendre, Gauss-Laguerre, Gauss-Hermite e Gauss-Chebyshev consoante o problema a resolver.

Um dos inconvenientes da quadratura Gaussiana é o facto dos seus argumentos e coeficientes serem, em geral, números irracionais, e por isso não são exactamente representados nos cálculos. Além disso, se a função integranda for conhecida apenas para um número determinado de pontos, estes podem não coincidir com os argumentos da fórmula Gaussiana.

Algoritmo 8.3 :

1. devem ser introduzidos os argumentos e coeficientes das diferentes fórmulas. Gauss-Legendre: $(y_i, B_i), i = 0, \ldots, 26$; Gauss-Laguerre: $(z_i, C_i), i = 0, \ldots, 26$; Gauss-Hermite: $(t_i, D_i), i = 0, \ldots, 26$
2. introduzir a função $f(x)$
3. ler a, b e n (< 7) e fazer $integ = 1$, $j = 1$

3.1. se $n = 2$ então fazer $j = 0$

3.2. senão, se $n = 6$ então fazer $j = 2$

3.3. calcular $k = int(\frac{n+1}{2})n + (-1)^n j$

4. se $b = \infty$ então

4.1. se $a = 0$ então

4.1.1. se confirma que $w(x) = e^{-x}$ então para $i = 0, \ldots, n$ fazer $x_i = z_{k+i}$ e $A_i = C_{k+i}$

4.1.2. senão terminar, não se aplica nenhuma das fórmulas

4.2. senão,

4.2.1. se $a = -\infty$ então

4.2.1.1. se confirma que $w(x) = e^{-x^2}$ então para $i = 0, \ldots, n$ fazer $x_i = t_{k+i}$ e $A_i = D_{k+i}$

4.2.1.2. senão terminar, não se aplica nenhuma das fórmulas

4.2.2. senão terminar, não se aplica nenhuma das fórmulas

5. senão,

5.1. se $a \neq -\infty$ então

5.1.1. se confirma que $w(x) = 1$ então para $i = 0, \ldots, n$ fazer $A_i = B_{k+i}$

5.1.1.1. se $a = -1$ e $b = 1$ então para $i = 0, \ldots, n$ fazer $x_i = y_{k+i}$

5.1.1.2. senão fazer $integ = \frac{b-a}{2}$ e para $i = 0, \ldots, n$ fazer $x_i = \frac{(b-a)y_{k+i}+a+b}{2}$

5.1.2. senão,

5.1.2.1. se confirma que $w(x) = \frac{1}{\sqrt{1-x^2}}$ então para $i = 0, \ldots, n$ fazer $A_i = \frac{\pi}{n+1}$

5.1.2.1.1. se $a = -1$ e $b = 1$ então para $i = 0, \ldots, n$ fazer $x_i = cos(\frac{(2i+1)\pi}{2n+2})$

5.1.2.1.2. senão fazer $integ = \frac{b-a}{2}$ e para $i = 0, \ldots, n$ fazer $\xi_i = cos(\frac{(2i+1)\pi}{2n+2})$ e $x_i = \frac{(b-a)\xi_i+a+b}{2}$

5.1.2.2. senão terminar, não se aplica nenhuma das fórmulas

5.2. senão terminar, não se aplica nenhuma das fórmulas

6. para $i = 0, \ldots, n$ calcular $f_i = f(x_i)$

7. calcular $integ = integ \sum_{i=0}^{n} A_i f_i$

8. terminar com $I \leftarrow integ$

8.4.3 Implementação. Rotina INTGAU

Na rotina INTGAU, que implementa o algoritmo 8.3, foram introduzidas tabelas com os coeficientes e argumentos das fórmulas de Gauss-Legendre, Gauss-Hermite e Gauss-Laguerre. Aceita, assim, problemas em que o número de subintervalos pode ir até seis.

Exemplo 8.5 O cálculo do integral

$$\int_{-0.25}^{0.25} e^x \, dx$$

pela fórmula de Gauss-Legendre com três pontos, faz uma transformação de variáveis por forma a passar do intervalo $[-0.25, 0.25]$ para $[-1, 1]$, usando a relação (8.21). O valor final obtido é 0.505225.

Exemplo 8.6 A rotina INTGAU, para a resolução de

$$\int_0^1 \sqrt{x}\, ln(x) \, dx$$

e considerando $n = 3$, ou seja quatro pontos, dá o valor -0.448878. Foi usada a fórmula de Gauss-Legendre.

Como a função integranda não é definida para o limite $x = 0$, não será possível resolver o integral usando uma das fórmulas compostas apresentadas em 8.2.3, a não ser que se use a técnica descrita no ponto 1. de 8.4.2.

Exemplo 8.7 A fórmula de Gauss-Laguerre para calcular

$$\int_0^\infty e^{-x}\, x^{1.2} \, dx$$

fornece, para quatro pontos, o valor 1.099323. Quando se usam três pontos o resultado obtido é 1.096904 e finalmente para dois pontos a aproximação conseguida é 1.088468.

8.5 Problemas

1. Calcule o integral

$$\int_0^1 f(x) \, dx$$

em que a função $f(x)$ é definida por

$$f(x) = \begin{cases} e^x & \text{se } 0 < x \leq 0.5 \\ sen(x) & \text{se } 0.5 < x \leq 1 \end{cases}$$

a) Usando um espaçamento entre pontos do intervalo constante e igual a 0.1.

b) Usando os seguintes pontos:

$$0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.7, 0.9, 1.0$$

Verifique neste caso a possibilidade de se usar a fórmula do trapézio, para os primeiros cinco subintervalos, a fórmula de Simpson, para o par seguinte de subintervalos e finalmente a fórmula do trapézio no subintervalo $[0.9, 1.0]$ e de o resultado ser 1.064430.

c) Calcule uma estimativa do erro de truncatura cometido com o valor obtido em a).

2. A fórmula de Newton-Cotes

$$\frac{b-a}{90}[7f(a) + 32f(\frac{3a+b}{4}) + 12f(\frac{a+b}{2}) + 32f(\frac{a+3b}{4}) + 7f(b)]$$

para aproximar o integral $I = \int_a^b f(x)dx$ é conhecida por fórmula de Milne. O seu erro de truncatura é dado por

$$e_M = -\frac{8(b-a)^7}{15482880} f^{vi}(\xi) \text{ com } \xi \in [a, b].$$

Deduza a correspondente fórmula composta e o respectivo erro de truncatura.

3. Um carro inicia um percurso, num dia frio de Inverno, e um aparelho mede o consumo de gasolina no instante em que percorreu $x\,km$. Os resultados registados são:

x, km	0.00	1.25	2.50	3.75	5.00	6.25	7.50	8.75	10.00
$f(x)$, l/km	0.260	0.208	0.172	0.145	0.126	0.113	0.104	0.097	0.092

em que $f(x)$ designa o consumo, naquele instante, em l/km. Calcule o consumo total de gasolina no fim do percurso de $10km$. Justifique a utilização da fórmula que usou. Calcule uma estimativa do erro de truncatura cometido.

4. Calcule a melhor aproximação ao integral

$$\int_0^{1.4} f(x)\,dx.$$

sendo a função $f(x)$ dada por:

x_i	0.0	0.1	0.2	0.3	0.45	0.6	0.75	0.9	1.0	1.1	1.2	1.3	1.4
$f(x_i)$	0	1	2	3	2	1	0	1	2	3	2	1	0

Estime o erro de truncatura cometido no intervalo $[0.9, 1.4]$.

5. Dada a tabela de valores de uma função $f(x)$

x_i	0.0	0.1	0.2	0.3	0.4	0.5	0.8	1.1	1.4	1.7
$f(x_i)$	0	1	1	2	2	3	3	4	4	5

   a) Calcule a melhor aproximação ao integral

$$\int_{0.0}^{1.7} f(x)dx$$

   usando todos os pontos da tabela.

   b) Estime o erro de truncatura cometido na alínea anterior.

   c) Calcule novamente o integral, usando apenas uma fórmula composta e um espaçamento constante igual a 0.3.

   d) Estime o erro de arredondamento cometido na alínea c.

6. Dado o integral

$$\int_{0.0}^{0.8} x\, e^x\, dx$$

   e a partir dos seguintes valores de x:

x_i	0.0	0.1	0.15	0.2	0.25	0.3	0.4	0.5	0.6	0.67	0.7	0.8

   a) Calcule, usando as fómulas de Newton-Cotes e respectivas fórmulas compostas, a melhor aproximação ao integral.

   b) Calcule novamente uma aproximação ao integral, considerando um espaçamento, h, constante entre pontos igual a 0.1.

   c) Estime o erro de truncatura cometido no intervalo $[0.0, 0.8]$ com a integração da alínea b.

7. Dado o integral

$$\int_{0.0}^{1.4} x\, sen(x)\, dx$$

   e a partir dos seguintes valores de x:

x_i	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.2	1.3	1.4

   a) Calcule, usando as fómulas de Newton-Cotes e respectivas fórmulas compostas, a melhor aproximação ao integral.

b) Estime o erro de truncatura cometido no intervalo $[0.0, 0.9]$ com a integração da alínea a.

8. Dada a função $S(x)$ definida por

$$S(x) = \int_0^x \frac{sen(t)}{t}\, dt$$

calcule $S(1)$

a) Usando a fórmula de Simpson e um espaçamento entre pontos do intervalo constante e igual a 0.0625. Considere

$$\frac{sen(0)}{0} = 1.$$

b) Usando a mesma fórmula de a), mas utilizando apenas dois subintervalos em $[0, 1]$. Verifique que o resultado obtido coincide com o de a) em três algarismos.

c) Usando a técnica de Romberg e a fórmula do trapézio, para os seguintes valores de h: $1, 0.5, 0.25, 0.125$ e 0.0625. Qual a precisão dos valores calculados, em termos de erros de truncatura?

d) Compare o resultado obtido pela técnica de Romberg, baseado nos valores de $h_1 = 0.125$ e $h_2 = 0.0625$, com o de a). Comente.

9. Dada a seguinte tabela de valores de $f(x)$

x	0.0	0.2	0.4	0.6	0.8	1.0	1.2	1.4	1.6
$f(x)$	1.000	1.221	1.492	1.822	2.226	2.718	3.320	4.056	5.216

calcule $\int_{0.0}^{1.6} f(x)dx$ usando a técnica de Romberg até obter uma aproximação que tenha um erro de truncatura da ordem de h^6.

10. Deduza as expressões para as aproximações ao integral $\int_a^b f(x)\, dx$, obtidas pela técnica de Romberg, quando implementada a partir da fórmula composta de Simpson.

11. Utilizando quadratura Gaussiana, calcule

$$\int_1^2 e^{x^2}\, dx$$

usando três pontos do intervalo. Repita o cálculo, mas agora com cinco pontos. Estime o erro de truncatura do primeiro valor calculado.

12. Calcule o integral

$$I = \int_0^1 \frac{sen(x)}{1+x}\, dx$$

usando a fórmula de Gauss-Legendre, baseada em seis pontos. Repita os cálculos usando agora quatro pontos do intervalo $[0, 1]$. Verifique que ambos os resultados coincidem, nas cinco primeiras casas decimais, com 0.28422.

13. Use quadratura Gaussiana para calcular o integral

$$\int_0^\infty e^{-x}(cos(x^2))^2\, dx$$

baseando os seus cálculos em quatro pontos.

14. Calcule o integral

$$\int_0^1 \frac{sen(x)}{\sqrt{1-x^2}}\, dx$$

usando a fórmula mais adequada e considerando três pontos no intervalo. Comente a escolha efectuada.

Capítulo 9

Equações diferenciais ordinárias

9.1 Introdução

9.1.1 Forma geral do problema

Em muitos problemas de engenharia a relação entre a variável independente, x, e a dependente, y, vem expressa em termos de uma equação diferencial[1]. Por exemplo, a equação que descreve a queda livre de um corpo, de massa m, num campo cuja aceleração da gravidade é g, e que relaciona o deslocamento, x, com o tempo t, é

$$m\frac{d^2x(t)}{dt^2} = -mg.$$

A solução analítica é $x(t) = -\frac{1}{2}gt^2 + c_1t + c_2$, com c_1 e c_2 constantes de integração calculadas em função das condições iniciais do sistema.

Outro exemplo de equação diferencial relaciona a velocidade, v, de um corpo em queda livre, com o tempo,

$$\frac{dv(t)}{dt} = g - \frac{c}{m}v(t)$$

em que c é o coeficiente de atrito. A equação é função de $v(t)$, da primeira derivada $v'(t)$ e de t (directa ou indirectamente). A quantidade a ser diferenciável v é a variável dependente; a quantidade a respeito da qual v é diferenciável, t, é a variável independente.

Genericamente, tem-se como forma geral de uma equação diferencial,

$$F(x, y(x), y'(x), y''(x), \ldots, y^{(n-1)}(x), y^{(n)}(x)) = 0$$

em que, a variável independente é, neste caso, x e a variável dependente, $y(x)$. A ordem mais elevada da derivada de $y(x)$ que surge na equação define a *ordem da equação diferencial*. O primeiro exemplo apresentado em cima, define uma equação de segunda ordem e o

[1] A necessidade da resolução numérica de equações diferenciais surgiu com os trabalhos desenvolvidos por Fourier, na Teoria do Calor, e por Adams, Bessel, Cauchy, Gauss, Lagrange, Laplace, Legendre, Leverrier, Poincaré e outros, na Mecânica Celeste, durante os séculos XVIII e XIX (Goldstine (1977)).

segundo, uma equação de primeira ordem. A forma geral pode ser colocada explicitamente em ordem à derivada de maior ordem,

$$y^{(n)}(x) = f(x, y(x), y'(x), \ldots, y^{(n-1)}(x)).$$

Quando existe apenas uma variável independente a equação designa-se por *equação diferencial ordinária*. Os dois exemplos apresentados são deste tipo. Quando estão envolvidas duas ou mais variáveis independentes, a equação designa-se por *equação diferencial com derivadas parciais*. A equação de Poisson

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(x, y)$$

em que $u = u(x, y)$, que descreve a distribuição da temperatura ao longo de uma placa, é um exemplo de equação diferencial com derivadas parciais.

9.1.2 Características do problema

Uma primeira classificação das equações diferenciais ordinárias diz respeito à ordem. A equação diferencial diz-se de *primeira ordem*, se estão envolvidas na equação a função e a sua primeira derivada. A forma geral, é pois

$$y'(x) = f(x, y(x)).$$

A equação diz-se de *segunda ordem*, se estão envolvidas a função e as primeira e segunda derivadas da função, que na forma geral explícita é

$$y''(x) = f(x, y(x), y'(x)).$$

A equação diz-se de *ordem superior* se estão envolvidas a função e as primeira, segunda e outras derivadas de ordem superior a dois.

Considere-se, agora, a seguinte equação diferencial

$$y'(x) = cos(x)$$

cuja solução analítica é

$$y(x) = \int cos(x)dx = sen(x) + c$$

em que c é a constante de integração. O problema, tal como foi definido, não tem uma solução única. Tudo depende do valor de c. Algumas soluções são apresentadas na figura 9.1. A curva a cheio corresponde a $c = 0$, a curva a tracejado a $c = 1$ e a curva ponteada a $c = -1$.

Adicionando condições auxiliares à equação, por forma a definir um valor para a constante de integração, a solução do problema passa a ser única. Assim, se para $x = 0$ a função tomar o valor 0, $y(0) = 0$, obtém-se $c = 0$. A solução do problema

$$y'(x) = cos(x)$$

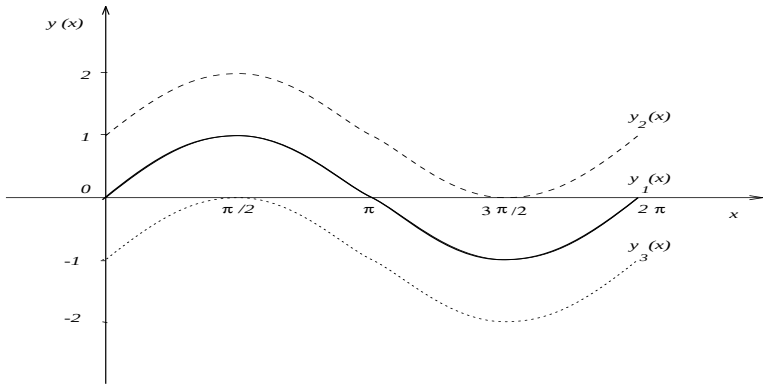


Figura 9.1: Soluções da equação $y'(x) = cos(x)$

no intervalo $[0, 2\pi]$, com a *condição inicial*

$$y(0) = 0$$

é única e igual a $y(x) = sen(x)$. As condições auxiliares devem ser especificadas de acordo com as propriedades físicas do problema.

Se a equação diferencial é de ordem n, do tipo

$$y^{(n)}(x) = f(x, y(x), y'(x), \ldots, y^{(n-1)}(x)),$$

torna-se necessário especificar n condições auxiliares para que ela possa ter uma única solução.

Se todas as condições auxiliares forem especificadas para o mesmo valor da variável independente, designadamente para o início do intervalo em que se quer a solução, o problema designa-se por *problema de equações diferenciais ordinárias com condições iniciais*. O exemplo

$$y''(x) = xy'(x) + x^2y(x) + 1 \quad \text{em que} \quad y(0) = 1 \ \text{ e } \ y'(0) = 1,$$

para $0 \leq x \leq 1$ define um problema de equação diferencial ordinária de segunda ordem com condições iniciais. Problemas deste tipo são tratados em 9.2, 9.3, 9.4 e 9.5.

Se as condições auxiliares dizem respeito a diferentes valores da variável independente, designadamente para valores dos extremos do intervalo onde se pretende a solução, o problema passa a ser caracterizado como *problema de equações diferenciais ordinárias com condições de fronteira*. Em 9.6 é apresentado um tipo de método numérico para a resolução deste tipo de problemas. A equação

$$y''(x) = x^2y'(x) + y(x) \ \text{ em que } \ y(1) = 1 \ \text{ e } \ y(2) = 1,$$

para $1 \leq x \leq 2$, é um exemplo típico deste tipo de problema.

9.1.3 Tipos de métodos

Seja $y(x)$ a função *solução exacta* do problema

$$y^{(n)}(x) = f(x, y(x), y'(x), \ldots, y^{(n-1)}(x)), \quad a \leq x \leq b. \tag{9.1}$$

A *solução numérica* consiste numa sequência de valores

$$y_0, y_1, y_2, \ldots, y_{n-2}, y_{n-1}, y_n$$

e que são aproximações à função $y(x)$, nos pontos $x_0, x_1, x_2, \ldots, x_{n-2}, x_{n-1}, x_n$ do intervalo $[a, b]$. Estes pontos encontram-se, normalmente, igualmente espaçados, e a distância entre eles é constante e igual a h.

Os métodos numéricos para a resolução de equações diferenciais ordinárias são métodos de discretização e *convertem* a equação diferencial numa equação diferença. Esta discretização consegue-se substituindo as derivadas, que aparecem na equação diferencial, por aproximações em função dos valores da função $y(x)$, em pontos do intervalo. A equação resultante é algébrica e é conhecida por equação diferença.

Uma *equação diferença de ordem* n é uma equação da forma

$$g_i(y_{i+n}, y_{i+n-1}, y_{i+n-2}, ..., y_i) = 0, \tag{9.2}$$

função de $n+1$ valores de y. Explicitando a equação em ordem ao y com o índice mais elevado, y_{i+n}, conclui-se, que este pode ser calculado em função de n valores de y, de ordem menor. Por isso a equação diferença se diz de ordem n.

Para os diferentes valores de i, $i = 0, 1, 2, ...$, conseguem-se calcular todos os valores que constituem a solução numérica, cobrindo todo o intervalo $[a, b]$. Para $i = 0$, a primeira aplicação de (9.2), obriga ao conhecimento de n valores de y: $y_0, y_1, y_2, \ldots, y_{n-1}$. Estes valores são chamados *valores auxiliares* e devem ser valores conhecidos, $y_k = c_k$ para $k = 0, 1, ..., n-1$.

Um caso particular de (9.2), que ocorre com alguma frequência, é o seguinte

$$\alpha_n y_{n+i} + \alpha_{n-1} y_{n+i-1} + \alpha_{n-2} y_{n+i-2} + ... + \alpha_0 y_i = 0 \tag{9.3}$$

para $i = 0, 1, 2, ...$ com $y_k = c_k$, para $k = 0, 1, ..., n-1$. As funções g_i, neste caso, são independentes de i, são funções homogéneas e lineares em y. Os coeficientes α_i são funções constantes. Chama-se *equação diferença linear e homogénea com coeficientes constantes.*

Assumiremos, sempre, que $\alpha_n \neq 0$ para que a equação seja de ordem n. Três exemplos muito simples de equações diferença são:

1. $$y_{i+1} - y_i = 0$$

   com $y_0 = 1$, que é de primeira ordem, uma vez que a aproximação y_{i+1} se obtém, da equação, a partir da informação relativa a um só ponto anterior, (x_i, y_i). Explicitando em ordem a y_{i+1}, tem-se $y_{i+1} = y_i$.

2.

$$2y_{i+2} - 17y_{i+1} + 8y_i = 0$$

com $y_0 = 2$ e $y_1 = 1$, que é de segunda ordem uma vez que y_{i+2} se obtém, da equação, a partir da informação relativa a dois pontos anteriores : (x_i, y_i) e (x_{i+1}, y_{i+1}).

3.

$$y_{i+3} - 2y_{i+2} - y_{i+1} + 2y_i = 0$$

com $y_0 = 0, y_1 = -3$ e $y_2 = 1$ e que é de ordem três.

9.1.4 Índice de algoritmos

São cinco os algoritmos desenvolvidos e apresentados neste Capítulo de equações diferenciais.

Algoritmo 9.1 Solução de uma equação diferencial de $1^{a\cdot}$ ordem, com condições iniciais, pelas fórmulas de Runge-Kutta de $1^{a\cdot}$, $2^{a\cdot}$, $3^{a\cdot}$ ou $4^{a\cdot}$ ordens.

Algoritmo 9.2 Solução de uma equação diferencial de $1^{a\cdot}$ ordem, com condições iniciais, pelas fórmulas preditoras-correctoras de Adams de $2^{a\cdot}$ ou $4^{a\cdot}$ ordens.

Algoritmo 9.3 Solução de um sistema de equações diferenciais de $1^{a\cdot}$ ordem, com condições iniciais, pelas fórmulas de Runge-Kutta de 2^a ordem ou preditora-correctora de Adams de $2^{a\cdot}$ ordem.

Algoritmo 9.4 Solução de uma equação ou sistema de equações diferenciais lineares de $1^{a\cdot}$ ordem 'stiff', pela fórmula de Gear.

Algoritmo 9.5 Solução de uma equação diferencial linear de $2^{a\cdot}$ ordem, com condições de fronteira definidas por y e/ou $1^{a\cdot}$ derivada, pelo método das diferenças finitas.

Para a resolução de uma equação diferencial de primeira ordem, os dois primeiros algoritmos são os indicados. No primeiro pode-se seleccionar um método de passo único e no segundo algoritmo a selecção é feita entre dois esquemas preditores-correctores de passo múltiplo. O algoritmo 9.3 é para a resolução de um sistema de n equações diferenciais de primeira ordem, o algoritmo 9.4 resolve uma ou um sistema de equações diferenciais 'stiff' e o 9.5 é específico para problemas com condições de fronteira.

9.2 Problemas com condições iniciais

9.2.1 Introdução teórica

Comecemos por estudar o caso mais simples de uma *equação diferencial de primeira ordem*. A forma geral do problema de uma equação diferencial ordinária de primeira ordem é

$$\frac{dy(x)}{dx} = f(x, y) \quad \text{com} \quad y(a) = y_0 \tag{9.4}$$

para $a \leq x \leq b$. O único valor auxiliar necessário para calcular a solução é dado para o início do intervalo, $x = a$, e o problema é caracterizado como sendo um problema de uma equação diferencial de primeira ordem com condições iniciais.

Como já foi referido em 9.1.3, os métodos numéricos discretizam a equação diferencial transformando-a numa equação diferença. É, assim, necessário, definir os pontos, do intervalo $[a, b]$, onde se vai calcular a solução numérica. Seja h o espaçamento constante entre esses pontos. Tem-se então

$$a = x_0 < x_1 < x_2 < x_3 < \ldots < x_{n-2} < x_{n-1} < x_n = b$$

com $h = x_{i+1} - x_i$, para $i = 0, 1, \ldots, n-1$.

9.2.2 Métodos de passo único. Fórmulas de Euler e de Runge-Kutta

Os métodos numéricos de passo único definem equações diferença de primeira ordem. Recordar o que foi referido em 9.1.3.

Se, no ponto de partida, (x_0, y_0), se calcular o valor do declive da tangente à curva $y(x)$, tem-se, a partir de (9.4)

$$y'(x_0) = f(x_0, y_0).$$

A partir desta informação pode-se calcular uma aproximação à função $y(x)$, no ponto seguinte do intervalo, x_1, e que vai ser designada por y_1, usando a equação diferença

$$y_1 = y_0 + hf(x_0, y_0)$$

sendo $h = x_1 - x_0$, o espaçamento constante entre os pontos do intervalo. Esta fórmula define a *equação do método de Euler*[2].

A figura 9.2 mostra a representação gráfica deste primeiro passo.

Aplicando a equação do método de Euler, para o passo seguinte, obtém-se

$$y_2 = y_1 + hf(x_1, y_1).$$

[2]Euler nasceu em Basel, na Suiça, em 1707. Em 1727 foi trabalhar para a Academia Russa de Ciências de S. Petersburgo e lá ficou durante 14 anos. Regressou, mais tarde, em 1766, vindo da Academia de Berlim, e lá ficou até à sua morte em 1783. Em 1771, devido a uma doença ficou cego, no entanto, não deixou de trabalhar nem de publicar. Metade dos seus trabalhos foram publicados a partir dessa altura. No total, publicou quase 600 trabalhos, entre livros e artigos, durante um período de quase 50 anos. É considerado um dos matemáticos notáveis do século XVIII. Foram inúmeras as suas contribuições, não só na Matemática como na Mecânica, Hidrodinâmica, Óptica e Astronomia. A célebre fórmula de Euler-MacLaurin foi apresentada por Euler em 1730 e, mais tarde, publicada por MacLaurin, em 1737. O método que aqui se descreve para a resolução de um problema de equações diferenciais com condições iniciais, foi proposto por Euler, em 1768. Também introduziu alguns símbolos matemáticos, hoje muito usados: e para base de logaritmos neperianos, $\imath$ para unidade imaginária $\sqrt{-1}$, Δy para diferenças finitas e $\sum$ para somatório. Em 1748 apresentou o valor de π com 127 casas decimais:

$$3,14159265358979323846264338327950288419716939937510582097494459230781640628620899$$

$$86280348253421170679821480865132823066470938446$$

(Goldstine (1977), Kahaner, Moler e Nash (1989) e Hämmerlin e Hoffmann (1991)).

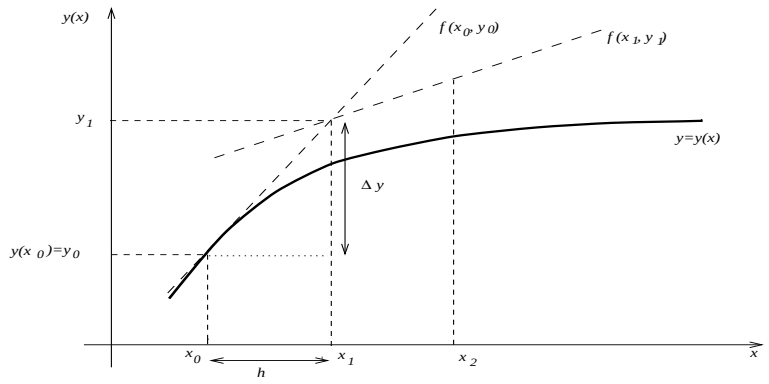


Figura 9.2: Representação gráfica do primeiro passo do método de Euler

Assim, para qualquer ponto x_i do intervalo, $i = 0, 1, \ldots, n-1$ a forma geral da equação de Euler é

$$y_{i+1} = y_i + hf(x_i, y_i), \quad i = 0, 1, \ldots \tag{9.5}$$

donde se conclui ser uma equação de *passo único*, uma vez que ela define uma equação diferença de primeira ordem (veja-se a equação (9.2)). É também caracterizada por ser uma fórmula *explícita*, uma vez que a solução y_{i+1}, $i = 0, 1, \ldots$ pode ser calculada recursivamente usando a equação (9.5).

Esta aproximação y_{i+1} de $y(x)$, no ponto $x = x_{i+1}$, tem erro de truncatura local da ordem de h^2, $\mathcal{O}(h^2)$, e o método diz-se de primeira ordem.

A expansão em série de Taylor de $y(x_{i+1})$,

$$y(x_{i+1}) = y(x_i) + hy'(x_i) + \frac{1}{2}h^2 y''(\xi_i)$$

em que ξ_i é um ponto do intervalo definido por x_i e x_{i+1}, usando a equação (9.4), pode escrever-se na seguinte forma

$$y(x_{i+1}) = y(x_i) + hf(x_i, y(x_i)) + \frac{1}{2}h^2 y''(\xi_i). \tag{9.6}$$

Se esta série for truncada após o segundo termo do lado direito, tem-se uma aproximação a $y(x_{i+1})$,

$$y_{i+1} = y(x_i) + hf(x_i, y(x_i))$$

que é a que se obtém pela equação de Euler. Assim, o termo truncado da série de Taylor define o erro de truncatura cometido com a aproximação obtida pela equação (9.5),

$$e_T = \frac{1}{2}h^2 y''(\xi_i) \quad \text{para} \quad \xi_i \in [x_i, x_{i+1}]. \tag{9.7}$$

O tamanho do erro de truncatura está relacionado com a precisão da aproximação calculada, y_{i+1}. É, pois, importante, saber-se estimar o erro de truncatura. Os algoritmos mais avançados para a resolução de equações diferenciais com condições iniciais, fazem uso de uma estimativa do erro de truncatura para ajustar o espaçamento entre os pontos x_i do intervalo $[a, b]$.

Notas sobre a convergência do método de Euler:

1. Se a curvatura da função $y(x)$ é muito acentuada, a aproximação y_{i+1}, $i = 0, 1, \ldots, n-1$, obtida pela equação Euler, depressa começa a divergir da solução exacta.

2. Se o espaçamento entre os pontos x_i e x_{i+1} for grande, a aproximação y_{i+1}, $i = 0, 1, \ldots, n-1$ cedo começa a divergir.

Pode-se melhorar a convergência, diminuindo o espaçamento entre os pontos. Se o espaçamento for reduzido para metade, o erro de truncatura é reduzido para um quarto.

Exemplo 9.1 Na resolução numérica de

$$y'(x) = cos(x)y(x) \text{ com } y(0) = 1,$$

no intervalo $0 \leq x \leq 2$, para um passo de $h = 0.5$, obtém-se

x_i	0.0	0.5	1.0	1.5	2.0
y_i	1.0	1.5	2.158187	2.741224	2.838177

pela fórmula de Euler. Para o erro local de truncatura

$$e_T = \frac{1}{2}h^2 y''(\xi_i) \text{ para } \xi_i \in [x_i, x_{i+1}] \ i = 0, \ldots, n-1,$$

como

$$y''(x) = f'(x, y(x)) = f'_y\, y'(x) + f'_x$$

com

$$y'(x) = cos(x)y(x), \quad f'_y = cos(x) \text{ e } f'_x = -sen(x)y(x),$$

tem-se

$$y''(x) = (cos^2(x) - sen(x))y(x)$$

e

$$e_T \leq \frac{1}{2}(0.5)^2 M_i.$$

M_i é o majorante de $y''(x)$ no intervalo $[x_i, x_{i+1}]$. Assim, para o primeiro passo $[x_0, x_1]$, $y''(x)$ é decrescente e toma valores entre $(cos^2(0) - sen(0))y_0 = 1$ e $(cos^2(0.5) - sen(0.5))y_1 \approx 0.4365$, donde se conclui que $M_0 = 1$. O limite superior do erro de truncatura cometido neste passo é de 0.125. O cálculo do erro para os passos seguintes segue o mesmo raciocínio.

As equações de Runge-Kutta[3] baseiam-se numa ideia semelhante à do método de Euler. A aproximação a $y(x)$, no ponto x_{i+1}, é calculada a partir de uma aproximação linear à curva, $y(x)$.

A mais simples de todas é a equação de Runge-Kutta de primeira ordem, que se obtém a partir da expansão em série de Taylor de $y(x_{i+1})$, ignorando os termos em função das derivadas de ordem superior ou igual a dois. A equação resultante é

$$y_{i+1} = y_i + hf(x_i, y_i)$$

e o erro local de truncatura é dado por

$$\frac{1}{2}h^2 y''(\xi_i)$$

para ξ_i do intervalo $[x_i, x_{i+1}]$ e que é a equação do método de Euler.

Usando o mesmo raciocínio, obtém-se outra aproximação a $y(x_{i+1})$, mas, desta vez, a expansão em série de Taylor é truncada após os termos que vêm em função das derivadas de ordem superior ou igual a três. A equação resultante é

$$y_{i+1} = y_i + hy'(x_i) + \frac{1}{2}h^2 y''(x_i)$$

ou

$$y_{i+1} = y_i + hf(x_i, y_i) + \frac{1}{2}h^2[f'_x(x_i, y_i) + f'_y(x_i, y_i)f(x_i, y_i)] \tag{9.8}$$

uma vez que

$$y''(x) = f'(x, y(x)) = f'_x + f'_y\, y'(x).$$

O erro de truncatura local é dado por

$$\frac{1}{3!}h^3 y'''(\xi_i)$$

com $\xi_i \in [x_i, x_{i+1}]$. A equação (9.8) vem em função das derivadas de f e por isso não é muito aconselhável. Em vez desta pode-se usar outra, que lhe seja equivalente. Considere esta fórmula na forma geral

$$y_{i+1} = y_i + \alpha_1 p + \alpha_2 q \tag{9.9}$$

[3]C.D.T. Runge (1856 - 1927) nasceu na Alemanha, foi estudante nas cidades de Munique e Berlim e fortemente influenciado por Weierstrass. De 1886 a 1904 esteve na Universidade Técnica de Hannover e posteriormente na Universidade de Göttinger. Runge considerava-se um matemático embora a maior parte do seu trabalho tenha sido desenvolvida em áreas aplicadas como a física, geodesia e astronomia. A sua preocupação, com a aplicação dos métodos numéricos a situações práticas, era tal, que uma das teses que defendeu, em Berlim em 1880, intitulava-se 'The value of a mathematical discipline depends on its applicability to the natural sciences'. Runge, juntamente com Fr. Willers, numa Enciclopédia que publicaram em 1915, falavam do trabalho notável de Euler na integração numérica das equações diferenciais ordinárias e parciais. O método aqui apresentado, para a resolução de uma equação diferencial, foi publicado, primeiro, em 1895, e mais tarde em 1924, num trabalho conjunto com H. König (Berlim). W. Kutta generalizou o resultado para um sistema de equações de primeira ordem, em 1901 (Goldstine (1977), Kahaner, Moler e Nash (1989) e Hämmerlin e Hoffmann (1991)).

em que

$$p = hf(x_i, y_i)$$

e

$$q = hf(x_i + \alpha_3 h, y_i + \alpha_3 p),$$

com α_1, α_2 e α_3 constantes a determinar. Substituindo p e q na equação (9.9) e usando a expansão em série de Taylor para $f(x_i + \alpha_3 h, y_i + \alpha_3 p)$, tem-se

$$y_{i+1} \approx y_i + \alpha_1 hf(x_i, y_i) + \alpha_2 h[f(x_i, y_i) + \alpha_3 hf'_x(x_i, y_i) + \alpha_3 pf'_y(x_i, y_i)]$$

ou ainda

$$\begin{aligned} y_{i+1} &\approx y_i + (\alpha_1 + \alpha_2)hf(x_i, y_i) + \alpha_2\alpha_3 h^2 f'_x(x_i, y_i) + \\ &+ \alpha_2\alpha_3 h^2 f(x_i, y_i) f'_y(x_i, y_i). \end{aligned}$$

Igualando os termos desta última equação com os de (9.8) obtém-se

$$\alpha_1 + \alpha_2 = 1$$

e

$$\alpha_2\alpha_3 = \frac{1}{2}.$$

Com duas equações e três incógnitas, arbitra-se o valor um para α_3, donde se conclui que $\alpha_1 = \alpha_2 = \frac{1}{2}$.

A fórmula final é então

$$y_{i+1} = y_i + \frac{1}{2}(p + q), \quad i = 0, 1, \ldots \tag{9.10}$$

com

$$p = hf(x_i, y_i)$$

e

$$q = hf(x_{i+1}, y_i + p),$$

e define a *equação do método de Runge-Kutta de segunda ordem*. Esta fórmula é *explícita* e de passo único.

A interpretação da variação Δy em (9.10), com

$$y_{i+1} = y_i + \Delta y,$$

é a de uma média pesada para o declive da tangente à curva, $y(x)$, no intervalo definido pelos pontos x_i e x_{i+1}. Esta média considera o declive da tangente à curva no ponto x_i e uma aproximação ao declive da tangente no ponto x_{i+1}.

Usando a igualdade

$$y''' = f''(x, y(x)) = f''_{xx} + 2f''_{xy}y'(x) + f''_{yy}(y'(x))^2 + f'_x f'_y + (f'_y)^2 y'(x)$$

obtém-se para a fórmula do erro de truncatura local da equação de Runge-Kutta de segunda ordem

$$e_T = \frac{1}{6}h^3[\frac{3}{2}(f''_{yy}(y'(x))^2 + 2f''_{xy}y'(x) + f''_{xx}) - f'_x - f'_y y'(x)]. \tag{9.11}$$

Confirma-se, assim, que a equação apresentada em (9.10) é de ordem dois.

Fórmulas de Runge-Kutta de ordens mais elevadas podem ser deduzidas de uma maneira semelhante às anteriores.

A *fórmula de Runge-Kutta de terceira ordem* tem a seguinte equação

$$y_{i+1} = y_i + \frac{1}{6}(p + 4q + r), \quad i = 0, 1, \ldots \tag{9.12}$$

com

$$p = hf(x_i, y_i),$$

$$q = hf(x_i + \frac{1}{2}h, y_i + \frac{1}{2}p)$$

e

$$r = hf(x_{i+1}, y_i - p + 2q).$$

O erro de truncatura local é $\mathcal{O}(h^4)$.

Das fórmulas mais populares, a equação do *método de Runge-Kutta de quarta ordem* é a que fornece melhores resultados. Embora não seja a mais simples de implementar, pois exige quatro cálculos de f em cada passo, tem duas propriedades importantes. Tem erro de truncatura local $\mathcal{O}(h^5)$ e é uma fórmula de passo único. Para calcular y_{i+1} apenas precisa da informação relativa ao ponto anterior (x_i, y_i),

$$y_{i+1} = y_i + \frac{1}{6}(p + 2q + 2r + s), \quad i = 0, 1, \ldots \tag{9.13}$$

com

$$p = hf(x_i, y_i),$$

$$q = hf(x_i + \frac{1}{2}h, y_i + \frac{1}{2}p),$$

$$r = hf(x_i + \frac{1}{2}h, y_i + \frac{1}{2}q)$$

e

$$s = hf(x_{i+1}, y_i + r).$$

Os valores p, q, r e s são estimativas da variação que se verifica em y, Δy, no intervalo $[x_i, x_{i+1}]$. Os valores de p e s fornecem estimativas que são baseadas no declive da tangente à curva $y(x)$ respectivamente no início, x_i, e no fim do intervalo, x_{i+1}. Por sua vez, q e r são estimativas da variação em y baseadas em aproximações ao declive da tangente à curva, $y(x)$, no ponto médio do intervalo, $x_i + \frac{1}{2}h$.

O algoritmo 9.1 implementa as quatro fórmulas de passo único da família das equações de Runge-Kutta. O parâmetro *ordem* indica a ordem da fórmula. Se for igual a 1, selecciona a equação (9.5), se for 2 selecciona (9.10), se for 3 selecciona (9.12) e finalmente, para o valor 4, escolhe (9.13).

Algoritmo 9.1 :

1. ler *ordem* (1,2,3 ou 4), h, x_0, y_0 e *sup* e calcular *tol*
2. introduzir a função $f(x,y)$
3. fazer $i = 0$
4. calcular $x_{i+1} = x_i + h$
5. se $(x_{i+1} - sup) \leq tol^{1/2}$ então
   - 5.1. calcular $f_i = f(x_i, y_i)$ e $p_1 = h\, f_i$
   - 5.2. se $ordem = 1$ então fazer $c_1 = 1$
   - 5.3. senão
     - 5.3.1. se $ordem = 2$ então fazer $c_1 = 1$, $c_2 = 1$ e calcular $z_i = y_i + p_1$, $g_i = f(x_{i+1}, z_i)$ e $p_2 = h\, g_i$
     - 5.3.2. senão
       - 5.3.2.1. se $ordem = 3$ então fazer $c_1 = 0.5$, $c_3 = c_1$ e $c_2 = 4c_1$ e calcular $z_i = y_i + 0.5p_1$, $r_i = x_i + 0.5h$, $g_i = f(r_i, z_i)$, $p_2 = h\, g_i$, $t_i = y_i - p_1 + 2p_2$, $g_i = f(x_{i+1}, t_i)$ e $p_3 = h\, g_i$
       - 5.3.2.2. senão fazer $c_1 = \frac{2}{3}$, $c_4 = c_1$, $c_2 = 2c_1$ e $c_3 = c_2$ e calcular $z_i = y_i + 0.5p_1$, $r_i = x_i + 0.5h$, $g_i = f(r_i, z_i)$, $p_2 = h\, g_i$, $z_i = y_i + 0.5p_2$, $g_i = f(r_i, z_i)$, $p_3 = h\, g_i$, $z_i = y_i + p_3$, $g_i = f(x_{i+1}, z_i)$ e $p_4 = h\, g_i$
   - 5.4. calcular $y_{i+1} = y_i + (\sum_{j=1}^{ordem} c_j p_j)/ordem$
   - 5.5. fazer $i = i + 1$, $n = i$ e ir para o passo 4
6. terminar com $y(x) \leftarrow (y_i, i = 0, 1 \ldots, n)$.

9.2.3 Implementação. Rotina EQUNIC

A implementação do algoritmo 9.1 origina a rotina EQUNIC.

Exemplo 9.2 A implementação dos métodos de passo único na resolução numérica da equação diferencial

$$y'(x) + y(x)^{1.5} - 1 = 0 \text{ com } y(0) = 10,$$

para $0 \leq x \leq 1$ origina os seguintes resultados:

x_i	$y_i(E)$	$y_i(R-K(2))$	$y_i(R-K(4))$
0.0	10.000000	10.000000	10.000000
0.1	6.937722	7.605179	7.537571
0.2	5.210357	5.992523	5.911206
0.3	4.121030	4.861635	4.784158
0.4	3.384448	4.042093	3.973540
0.5	2.861815	3.432006	3.373074
0.6	2.477685	2.967664	2.917574
0.7	2.187681	2.607674	2.565252
0.8	1.964105	2.324245	2.288313
0.9	1.788843	2.098174	2.067683
1.0	1.649589	1.915858	1.889916

O valor do passo considerado é de $h = 0.1$. A primeira coluna de y_i corresponde aos resultados obtidos pela fórmula de Euler, a segunda coluna ao método de Runge-Kutta de segunda ordem e a última ao método de Runge-Kutta de quarta ordem.

9.2.4 Métodos de passo múltiplo. Equação de Euler melhorada e fórmula de Adams-Bashforth

Dada a equação diferencial de primeira ordem

$$y'(x) = f(x, y(x))$$

com a condição inicial $y(x_0) = y_0$, suponha ser também possível calcular o valor de $y(x_1)$ ou uma aproximação, y_1, usando uma fórmula de passo único. Com esta informação adicional (x_1, y_1), é possível determinar o valor do declive da tangente à curva no ponto x_1, $y'(x_1) = f(x_1, y_1)$. Uma aproximação ao valor de $y(x)$ no ponto x_2 é então calculada através da fórmula conhecida por *equação de Euler melhorada*,

$$y_2 = y_0 + 2hf(x_1, y_1),$$

sendo h o espaçamento constante entre os pontos. No passo seguinte, calcula-se y_3 em função de y_2 e y_1 e assim sucessivamente. Esta fórmula pode obter-se a partir da aproximação à derivada $y'(x_{i+1})$, pela diferença central de primeira ordem

$$y'(x_{i+1}) \approx \frac{1}{2h}(y_{i+2} - y_i).$$

Assim, a equação geral do método é definida pela seguinte equação diferença de *passo duplo*

$$y_{i+2} = y_i + 2hf(x_{i+1}, y_{i+1}), \quad i = 0, 1, \ldots \tag{9.14}$$

O erro de truncatura local é dado por

$$e_T = \frac{1}{3}h^3 y'''(\xi_i) \tag{9.15}$$

com ξ_i do intervalo $[x_i, x_{i+2}]$.

Uma vez que se trata de uma fórmula de segunda ordem é aconselhável utilizar o método de Kunge-Kutta, de passo único, e também de segunda ordem para o cálculo do valor auxiliar y_1.

Outro exemplo de um método de *passo duplo* é o que corresponde à *fórmula de Adams-Bashforth*[4] *de segunda ordem*. A equação é *explícita* e na sua forma geral é

$$y_{i+2} = y_{i+1} + \frac{1}{2}h[3f(x_{i+1}, y_{i+1}) - f(x_i, y_i)], \quad i = 0, 1, \ldots \tag{9.16}$$

sendo o erro de truncatura local

$$e_T = \frac{5}{12}h^3 y'''(\xi_i) \tag{9.17}$$

com ξ_i do intervalo $[x_i, x_{i+2}]$. Tal como a fórmula (9.14), necessita da informação relativa ao ponto (x_1, y_1), além do valor inicial y_0 em $x = x_0$, para se poder implementar. Este valor auxiliar deve ser encontrado com a ajuda do método de Runge-Kutta de segunda ordem.

Existem outras fórmulas de Adams-Bashforth de ordem superior. A de ordem quatro é apresentada em (9.29) no âmbito de esquemas preditores-correctores para a resolução de equações diferenciais com condições iniciais.

9.2.5 Fórmula implícita. Equação de Adams-Moulton

O método de Adams-Moulton[5] *de ordem dois* é também conhecido por regra do trapézio. Até esta altura é a única equação *implícita* apresentada. A informação y_{i+1} referente ao ponto seguinte, x_{i+1}, surge também no membro do lado direito da equação, em $f(x, y)$. A forma geral é

$$y_{i+1} = y_i + \frac{1}{2}h[f(x_{i+1}, y_{i+1}) + f(x_i, y_i)], \quad i = 0, 1, \ldots \tag{9.18}$$

e embora seja de passo único, podem surgir algumas dificuldades quando da sua implementação. Se a equação diferencial for linear, torna-se fácil explicitar a informação relativa ao ponto seguinte, y_{i+1}. Por sua vez, se a equação diferencial for não linear, também a equação diferença do método é não linear e a sua resolução pode obrigar à utilização de

[4] J.C. Adams foi não só um astrónomo notável como um excelente homem de cálculos matemáticos. Calculou a constante de Euler com 515 algarismos (em 1878) e os primeiros sessenta e dois números de Bernoulli, com 110 algarismos (em 1877). As fórmulas de Adams-Bashforth, bem como as de Adams-Moulton combinadas em esquemas preditores-correctores parecem ser devidas apenas a Adams, embora apareçam referenciadas num trabalho de Bashforth (1883) (Goldstine (1977)).

[5] Esta fórmula de Adams foi melhorada por F.R. Moulton já no século XX. A técnica de Moulton consistia em usar uma combinação de um método que extrapolasse para obter uma primeira aproximação, usando uma equação de Adams, numa sequência de iterações de interpolação, para melhorar essa aproximação. Aliás, esta era a ideia dos esquemas de Adams. A diferença principal está na utilização, por parte de Moulton, de um certo tipo de fórmulas e na análise da sua convergência ('New Methods in Exterior Ballistics' (1925) e 'Differential Equations' (1930) em Goldstine (1977)).

um método iterativo, designadamente o método de Newton. Quando a equação não linear é uma forma polinomial de grau dois, a sua resolução é feita pela fórmula resolvente. Repare-se nos dois exemplos seguintes:

Exemplo 9.3 Usando a equação diferencial do exemplo 9.1,

$$y'(x) = cos(x)y(x) \text{ com } y(0) = 1$$

que é linear em $y(x)$, a fórmula de Adams-Moulton origina o seguinte esquema

$$y_{i+1} = y_i + \frac{1}{2}h[cos(x_i)y_i + cos(x_{i+1})y_{i+1}]$$

para $i = 0, 1, \ldots$ Explicitando e resolvendo em ordem a y_{i+1}, tem-se

$$y_{i+1}(1 - \frac{1}{2}hcos(x_{i+1})) = y_i(1 + \frac{1}{2}hcos(x_i))$$

ou

$$y_{i+1} = \frac{y_i(1 + \frac{1}{2}hcos(x_i))}{(1 - \frac{1}{2}hcos(x_{i+1}))}.$$

Considerando $h = 0.5$ e $i = 0$, obtém-se

$$y_1 = (1 + 0.25)/(1 - 0.25 \times 0.877582561) = 1.601323.$$

Todos os valores seguintes são retirados da mesma equação para $i = 1, 2, \ldots$

Exemplo 9.4 Vamos ver o que é que acontece com a equação

$$y'(x) = y(x)^2 \text{ com } y(0) = 0$$

que é não linear em $y(x)$. A fórmula de Adams-Moulton origina o esquema

$$y_{i+1} = y_i + \frac{1}{2}h[y_i^2 + y_{i+1}^2]$$

para $i = 0, 1 \ldots$. Resolvendo em ordem a y_{i+1}, fica-se com

$$y_{i+1} - \frac{1}{2}hy_{i+1}^2 = y_i(1 + \frac{1}{2}hy_i)$$

e como, por exemplo, para $i = 0$, se conhece $y_0 = 0$, obtém-se para $h = 0.25$

$$y_1 - \frac{1}{2}(0.25)y_1^2 = 0.$$

Esta equação algébrica é não linear em y_1,

$$y_1 - 0.125y_1^2 = 0$$

e tem de ser resolvida para se determinar o y_1, aproximação a $y(x_1)$ (no primeiro passo). Para o caso particular desta função $f(x, y(x))$, o cálculo de y_1 faz-se pela utilização da fórmula resolvente, uma vez que a equação é do segundo grau. No entanto, para a generalidade dos exemplos não lineares, esta fase necessita da utilização de um método iterativo, nomeadamente o de Newton, para calcular a raiz da equação algébrica não linear em y_{i+1}, para qualquer i.

O erro de truncatura local da fórmula de Adams-Moulton é

$$e_T = -\frac{1}{12}h^3y'''(\xi_i) \tag{9.19}$$

para ξ_i do intervalo definido pelos pontos x_i e x_{i+1}.

9.2.6 Erro de truncatura local. Erro global e estabilidade do método

A maior dificuldade que surge na resolução numérica das equações diferenciais reside na estimativa dos erros cometidos ao longo do processo numérico.

Se o valor de y_{i+1} é calculado por uma equação diferença a partir de valores *exactos* de $y(x)$: $y(x_i)$, $y(x_{i-1}), y(x_{i-2}), \ldots$, então, o erro no cálculo de y_{i+1},

$$e_T = y(x_{i+1}) - y_{i+1}$$

chama-se *erro de truncatura local* da equação.

O *erro global* de uma equação é o erro no valor calculado y_{i+1} quando todos os valores anteriores são os $y_i, y_{i-1}, y_{i-2}, \ldots$ e foram *também calculados* pela mesma equação diferença. Isto é, o erro global consiste na soma do erro local do passo, com a acumulação dos erros locais dos passos anteriores e dos erros de arredondamento. Para se ver bem a diferença entre estes erros, considere o caso do método de Euler aplicado à equação:

$$y'(x) = y(x)$$

para $0 \leq x \leq 1$, com $y(0) = 1$.

A solução exacta é $y(x) = e^x$.

A aproximação conseguida pelo método de Euler é a seguinte

$$y_{i+1} = y_i + hf(x_i, y_i) = (1 + h)y_i$$

para $i = 0, 1, 2, \ldots$, com $y_0 = 1$. Esta equação diferença é linear, e tem como solução $y_i = (1 + h)^i$, $i = 0, 1, 2, \ldots$ Assim, o erro global é

$$e_G = |y(x_i) - y_i| = |e^{x_i} - (1 + h)^i| = |e^{ih} - (1 + h)^i|$$

que pode ser aproximado por

$$e_G \approx \frac{1}{2}h(e^{x_i} - 1) \leq \frac{1}{2}h(e - 1) \quad \text{para} \quad 0 \leq x_i \leq 1.$$

Da fórmula do erro de truncatura do método de Euler (9.7), e como $y''(\xi) = e^{\xi}$, para $x_i \leq \xi_i \leq x_{i+1}$, tem-se

$$e_T = \frac{1}{2}h^2 e^{\xi_i} \leq \frac{1}{2}h^2 e.$$

Comparando este erro local, e_T, com o erro global, e_G, verifica-se que aquele é aproximadamente $1.5h$ vezes menor do que o erro global.

Neste exemplo foi fácil analisar o erro global porque a solução exacta da equação diferencial era conhecida. No entanto, na maior parte dos casos, ela é desconhecida. Normalmente torna-se mesmo impossível estimar o erro global com precisão. Pode-se, no entanto, considerar a relação qualitativa entre os erros local e global e usar a estimativa do erro local para determinar a ordem de grandeza do erro global.

Outra análise que pode tornar-se também um pouco complexa, é a análise da estabilidade das equações diferença.

Para uma equação diferencial de primeira ordem, seria suficiente uma fórmula numérica de passo único, conhecida por equação diferença de primeira ordem, uma vez que tanto a equação diferencial como a equação diferença têm uma única solução complementar.

A solução analítica da equação diferencial é composta por uma combinação de um integral particular e funções complementares, tantas quantas a ordem da equação,

$$y(x) = y_0(x) + A_1 y_1(x) + \cdots + A_s y_s(x).$$

Por exemplo, para esta equação de primeira ordem, linear com coeficientes constantes

$$y'(x) = ay(x) + b \tag{9.20}$$

a única função complementar da solução decresce exponencialmente se a for negativo e cresce exponencialmente se $a > 0$.

A solução do problema algébrico é também formada por uma solução particular e uma combinação de soluções complementares. Este número depende da ordem da equação diferença, ou do número de passos da fórmula numérica,

$$y_i = y_i^{(0)} + A_1 y_i^{(1)} + \cdots + A_s y_i^{(s)}. \tag{9.21}$$

Quando a equação diferença tem coeficientes constantes,

$$y_i^{(k)} = x_k^i$$

em que x_k é um zero do polinómio característico associado à equação diferença e $k = 1, \ldots, s$, sendo s a ordem da equação diferença.

Considere-se novamente uma equação diferencial de primeira ordem. A fórmula numérica, de passo único, considera-se 'boa' se a sua solução complementar é semelhante, no sentido de que tem o mesmo comportamento, à correspondente solução da equação diferencial. No entanto, a fórmula numérica pode ter mais do que uma solução complementar, se for de passo múltiplo e neste caso tem de se ter cuidado com o comportamento das soluções complementares parasitas. Estas *soluções parasitas* são as que não têm correspondente na solução da equação diferencial. Para equações de ordem superior aplica-se o mesmo raciocínio.

Quando o coeficiente constante, a, da equação (9.20) é negativo, a sua solução complementar decresce exponencialmente com x e as raízes x_k do polinómio característico devem verificar $|x_k| < 1$, por forma a que a solução y_i tenha o mesmo tipo de comportamento do de $y(x)$. Se o coeficiente é positivo, a solução complementar cresce exponencialmente e o problema do comportamento das soluções complementares parasitas não se torna grave, pois a sua influência não se faz sentir na solução final.

Em virtude disto, podem surgir três tipos de instabilidade induzida pelos métodos. A *instabilidade parcial* pode surgir nas equações diferença que têm o mesmo número de

soluções do que a equação diferencial e também nas equações com soluções parasitas. O método torna-se estável para um conjunto limitado de valores de h. O problema da instabilidade desaparece quando $h \to 0$.

Se as soluções parasitas dominam para qualquer valor de h e até pioram à medida que $h \to 0$, o método diz-se que induz *forte instabilidade* para qualquer tipo de problema.

Outro tipo de *fraca instabilidade* induzida pelas soluções parasitas é a que ocorre para qualquer valor de h embora só seja perigosa para alguns problemas.

Exemplo 9.5 Considere-se a equação diferencial linear em $y(x)$

$$y'(x) = -2y(x) + 1 \text{ com } y(0) = 1$$

em que o coeficiente constante é negativo e analisem-se as equações numéricas de três fórmulas:

- Euler, com erro de truncatura da ordem de h^2,

  $$y_{i+1\,E} = y_{i\,E} + h(-2y_{i\,E} + 1)$$

  para $i = 0, 1, \ldots$ Tem uma única solução complementar, que se obtém a partir de

  $$y_{i+1\,E} + y_{i\,E}(2h - 1) = h$$

  construindo o polinómio característico que lhe corresponde

  $$x + (2h - 1) = 0$$

  e que é de grau um. A correspondente raiz é $x = 1 - 2h$. A solução geral desta equação diferença é

  $$y_i = y_i^{(0)} + A_1(1 - 2h)^i$$

  e para $|x| < 1$, que é equivalente a $h < 1$, o método é estável. Como se verifica estabilidade e a solução da equação diferença também decresce à medida que x aumenta, para um conjunto limitado de valores de h, pode dizer-se que o método induz instabilidade parcial.

- Euler melhorada, com erro de truncatura menor do que o caso anterior, da ordem de h^3,

  $$y_{i+2\,Em} = y_{i\,Em} + 2h(-2y_{i+1\,Em} + 1)$$

  para $i = 0, 1, \ldots$ Tem duas soluções complementares, que se obtêm a partir de

  $$y_{i+2\,Em} + 4hy_{i+1\,Em} - y_{i\,Em} = 2h$$

  construindo o polinómio característico que lhe corresponde

  $$x^2 + 4hx - 1 = 0,$$

  de grau dois , donde se tiram as raízes $x_1 = -2h + \sqrt{4h^2 + 1} \approx 1 - 2h$ e $x_2 = -2h - \sqrt{4h^2 + 1} \approx -(1 + 2h)$. A solução geral desta equação diferença é

  $$y_i = y_i^{(0)} + A_1(1 - 2h)^i + (-1)^i A_2(1 + 2h)^i$$

  e a segunda solução complementar é parasita e tem-se $|1 + 2h| > 1$ para qualquer valor de $h \neq 0$. Este método induz fraca instabilidade.

- Adams-Mouton, com erro de truncatura da ordem de h^3,

$$y_{i+1\,AM} = y_{i\,AM} + \frac{1}{2}h(-2y_{i+1\,AM} + 1 - 2y_{i\,AM} + 1)$$

para $i = 0, 1, \ldots$ Tem uma solução complementar, que se obtém a partir de

$$y_{i+1\,AM}(1+h) + y_{i\,AM}(h-1) = h$$

e do polinómio característico correspondente

$$x(1+h) + h - 1 = 0$$

de grau um. A raiz é então

$$x = \frac{1-h}{1+h}$$

verificando-se $|x| < 1$ para qualquer valor de h positivo. Assim, o método implícito de Adams-Moulton é estável.

9.2.7 Esquemas preditores-correctores de Euler e Adams

Em 9.2.5 fez-se referência a uma fórmula *implícita* de segunda ordem. As fórmulas implícitas são, em geral, mais precisas do que as explícitas. Além disso, originam equações diferença quase sempre bem condicionadas. O grande inconveniente na sua utilização reside no facto do valor y_{i+1} estar definido implicitamente. Se a função $f(x, y(x))$ é não linear, o cálculo de y_{i+1} necessita da resolução de uma equação não linear.

Um *esquema preditor-corrector*, para a resolução numérica de uma equação diferencial, consiste na implementação de duas fórmulas, sendo uma *explícita* e a outra *implícita*.

Em cada ponto x_{i+2}, do intervalo onde se quer a solução, a fórmula explícita é usada, em primeiro lugar, para *prever* o valor de y_{i+2}. O valor correspondente designa-se por y^p_{i+2} e a equação diferença chama-se *fórmula preditora*. Este valor é depois corrigido, utilizando-o no membro do lado direito de uma fórmula implícita. Esta fórmula chama-se *correctora* e o valor que daqui se determina, y^c_{i+2}, é a melhor aproximação calculada a $y(x)$, no ponto x_{i+2}.

As fórmulas usadas como preditora e correctora, num esquema preditor-corrector, devem ser da mesma ordem. O esquema mais simples, é o esquema *preditor-corrector de Euler* que usa como fórmula preditora, a equação de Euler melhorada de passo duplo,

$$y^p_{i+2} = y_i + 2hf(x_{i+1}, y_{i+1}) \tag{9.22}$$

e como fórmula correctora, a equação de Adams-Moulton, de passo único

$$y^c_{i+2} = y_{i+1} + \frac{1}{2}h[f(x_{i+1}, y_{i+1}) + f(x_{i+2}, y^p_{i+2})] \tag{9.23}$$

para $i = 0, 1, \ldots$

Os erros de truncatura (9.15) da fórmula (9.22), e (9.19) de (9.23), podem ser escritos na forma

$$y^p_{i+2} - y(x_{i+2}) = \tau^p_i h^3 + \mathcal{O}(h^4)$$

e

$$y_{i+2}^c - y(x_{i+2}) = \tau_i^c h^3 + \mathcal{O}(h^4)$$

com

$$\tau_i^p = \frac{1}{3}y'''(x_i) \quad \text{e} \quad \tau_i^c = -\frac{1}{12}y'''(x_i).$$

Explicitando h^3 da primeira e substituindo na segunda fórmula do erro, fica-se com

$$y_{i+2}^c - y(x_{i+2}) \approx \tau_i^c \frac{y_{i+2}^p - y(x_{i+2})}{\tau_i^p}$$

ou

$$y_{i+2}^c - y(x_{i+2}) \approx \frac{\tau_i^c}{\tau_i^p - \tau_i^c}[y_{i+2}^p - y_{i+2}^c]. \tag{9.24}$$

Usando os valores de τ_i^p e τ_i^c, definidos anteriormente, nesta expressão, obtém-se para o erro local do valor corrigido obtido pelo esquema preditor-corrector de Euler

$$y_{i+2}^c - y(x_{i+2}) \approx -\frac{1}{5}[y_{i+2}^p - y_{i+2}^c,] \tag{9.25}$$

em função dos valores y_{i+2}^p e y_{i+2}^c.

O esquema preditor-corrector mais popular é o de *Adams de segunda ordem* que usa a equação de Adams-Bashforth de segunda ordem como fórmula preditora e como correctora, a de Adams-Moulton de segunda ordem. Assim, o esquema é definido pelas equações

$$y_{i+2}^p = y_{i+1} + \frac{1}{2}h[3f(x_{i+1}, y_{i+1}) - f(x_i, y_i)] \tag{9.26}$$

e

$$y_{i+2}^c = y_{i+1} + \frac{1}{2}h[f(x_{i+1}, y_{i+1}) + f(x_{i+2}, y_{i+2}^p)] \tag{9.27}$$

para $i = 0, 1, \ldots$.

Seguindo uma dedução idêntica à usada no esquema anterior, e como as fórmulas usadas (9.26) e (9.27) têm erros locais respectivamente iguais a $\frac{5}{12}h^3y'''(\xi_i)$ e $-\frac{1}{12}h^3y'''(\eta_i)$, a partir da expressão (9.24), e como para este caso

$$\tau_i^p = \frac{5}{12}y'''(x_i) \quad \text{e} \quad \tau_i^c = -\frac{1}{12}y'''(x_i),$$

tem-se a seguinte aproximação ao erro local do valor corrigido do esquema preditor-corrector de Adams de ordem dois,

$$y_{i+2}^c - y(x_{i+2}) \approx -\frac{1}{6}[y_{i+2}^p - y_{i+2}^c]. \tag{9.28}$$

Um dos inconvenientes dos esquemas preditores-correctores que usam fórmulas de passo múltiplo, reside na necessidade de calcular um ou mais valores auxiliares que, conjuntamente com o valor dado pela condição inicial (y_0), são necessários para a sua implementação.

O número desses valores auxiliares depende do número de passos das fórmulas preditora e correctora. Devem, no entanto, ser calculados a partir de uma fórmula de passo único que gere aproximações com erros locais da mesma ordem da do método preditor-corrector. Valores auxiliares calculados com erros de truncatura superiores aos que se obtêm com o esquema preditor-corrector, originam um mau aproveitamento do esquema, que poderia fornecer resultados mais precisos e não o faz porque os valores de partida vêm já com pouca precisão. Por sua vez, valores auxiliares com grande precisão acabam por ser mal aproveitados, uma vez que os valores seguintes não terão a mesma precisão. Um procedimento generalizado para o cálculo destes valores auxiliares baseia-se na utilização dos métodos de Runge-Kutta da ordem mais adequada.

Para o esquema preditor-corrector de Adams de segunda ordem o método de Runge-Kutta de segunda ordem é o mais indicado.

Trabalhoso mas fornecendo resultados bastante satisfatórios é o *esquema preditor-corrector de Adams de quarta ordem*. Usa, como fórmula preditora a *equação explícita de Adams-Bashforth de quarta ordem e de quatro passos*,

$$y^p_{i+4} = y_{i+3} + \frac{1}{24}h[55f(x_{i+3}, y_{i+3}) - 59f(x_{i+2}, y_{i+2}) + 37f(x_{i+1}, y_{i+1}) - 9f(x_i, y_i)] \quad (9.29)$$

cujo erro local é

$$e_T = \frac{251}{720}h^5 y^{(v)}(\xi_i), \quad \xi_i \in [x_i, x_{i+4}] \quad (9.30)$$

e como fórmula correctora a *equação implícita de Adams-Moulton de ordem quatro e de três passos*

$$y^c_{i+4} = y_{i+3} + \frac{1}{24}h[9f(x_{i+4}, y^p_{i+4}) + 19f(x_{i+3}, y_{i+3}) - 5f(x_{i+2}, y_{i+2}) + f(x_{i+1}, y_{i+1})], \quad (9.31)$$

para $i = 0, 1, \ldots$ O erro de truncatura local da fórmula correctora é

$$e_T = -\frac{19}{720}h^5 y^{(v)}(\eta_i), \quad \eta_i \in [x_{i+1}, x_{i+4}]. \quad (9.32)$$

Usando a expressão (9.24) para estimar o erro local do valor calculado pela fórmula correctora no esquema, a partir do valor previsto, tem-se

$$y^c_{i+4} - y(x_{i+4}) \approx -\frac{19}{270}(y^p_{i+4} - y^c_{i+4}). \quad (9.33)$$

Os esquemas preditores-correctores de Adams de segunda e quarta ordens são os métodos, de passo múltiplo, apresentados no algoritmo 9.2. O parâmetro *ordem* selecciona a ordem do esquema preditor-corrector de Adams que se quer implementar.

Algoritmo 9.2 :

1. `ler` *ordem* `(2 ou 4),` h`,` x_0`,` y_0 `e` *sup* `e calcular` *tol*
2. `introduzir a função` $f(x, y)$

3. para $j = 0, ordem - 2$

   3.1. calcular $x_{j+1} = x_j + h$

   3.2. calcular $f_j = f(x_j, y_j)$ e $p_1 = h\, f_j$

      3.2.1. se $ordem = 2$ então fazer $c_1 = 1$, $c_2 = 1$ e calcular $z_j = y_j + p_1$, $g_j = f(x_{j+1}, z_j)$ e $p_2 = h\, g_j$

      3.2.2. senão fazer $c_1 = \frac{2}{3}$, $c_4 = c_1$, $c_2 = 2c_1$ e $c_3 = c_2$ e calcular $z_j = y_j + 0.5p_1$, $r_j = x_j + 0.5h$, $g_j = f(r_j, z_j)$, $p_2 = h\, g_j$, $z_j = y_j + 0.5p_2$, $g_j = f(r_j, z_j)$, $p_3 = h\, g_j$, $z_j = y_j + p_3$, $g_j = f(x_{j+1}, z_j)$ e $p_4 = h\, g_j$

   3.3. calcular $y_{j+1} = y_j + (\sum_{k=1}^{ordem} c_k p_k)/ordem$

4. fazer $i = j + 1$

5. calcular $x_{i+1} = x_i + h$

6. se $(x_{i+1} - sup) \leq tol^{1/2}$ então

   6.1. calcular $f_i = f(x_i, y_i)$

   6.2. se $ordem = 2$ então calcular $y_{i+1} = y_i + \frac{h}{2}(3f_i - f_{i-1})$, $f_{i+1} = f(x_{i+1}, y_{i+1})$ e $y_{i+1} = y_i + \frac{h}{2}(f_{i+1} + f_i)$

   6.3. senão calcular $y_{i+1} = y_i + \frac{h}{24}(55f_i - 59f_{i-1} + 37f_{i-2} - 9f_{i-3})$, $f_{i+1} = f(x_{i+1}, y_{i+1})$ e $y_{i+1} = y_i + \frac{h}{24}(9f_{i+1} + 19f_i - 5f_{i-1} + f_{i-2})$

   6.4. fazer $i = i + 1$, $n = i$ e ir para o passo 5

7. terminar com $y(x) \leftarrow (y_i, i = 0, 1 \ldots, n)$.

9.2.8 Implementação. Rotina EQUDIF

A rotina EQUDIF surge da implementação do algoritmo 9.2, e resolve uma equação diferencial de primeira ordem, com condições iniciais, utilizando os esquemas preditores-correctores de Adams de segunda ou quarta ordens.

Exemplo 9.6 Na resolução numérica da equação diferencial do exemplo 9.2 e tomando $h = 0.1$ obtêm-se, pelo esquema preditor-corrector de Adams de segunda ordem, os resultados da primeira coluna y_i da tabela:

x_i	$y_i(P.C.(A-2))$	$y_i(P.C.(A-4))$
0.0	10.000000	10.000000
0.1	-	-
0.2	5.895740	-
0.3	4.738120	-
0.4	3.921412	3.967272
0.5	3.323503	3.365837
0.6	2.873256	2.910985
0.7	2.526658	2.559465
0.8	2.255096	2.283397
0.9	2.039236	2.063558
1.0	1.865594	1.886466

Uma vez que a fórmula preditora de Adam-Bashforth de segunda ordem é de dois passos, houve necessidade de recorrer à fórmula de Runge-Kutta de ordem dois para calcular o valor de y_1. O valor calculado foi 7.605179, que aparece na 2^a linha da 2^a coluna y_i da tabela do exemplo 9.2. Para o esquema preditor-corrector de Adams de quarta ordem, há necessidade de calcular y_1, y_2 e y_3 pela fórmula de Runge-Kutta da mesma ordem. Assim, considerando os valores 7.537571, 5.911206 e 4.784158, presentes na 3^a coluna y_i da tabela do exemplo 9.2, os restantes resultados obtidos pela preditora-correctora de Adams de ordem quatro são os apresentados na última coluna desta tabela.

9.3 Sistemas de equações diferenciais

9.3.1 Introdução

Um *sistema de n equações diferenciais de primeira ordem* tem a seguinte forma geral

$$\begin{cases} y_1'(x) = f_1(x, y_1(x), y_2(x), ..., y_n(x)) \\ y_2'(x) = f_2(x, y_1(x), y_2(x), ..., y_n(x)) \\ \dots \\ y_n'(x) = f_n(x, y_1(x), y_2(x), ..., y_n(x)) \end{cases}$$

para $a \leq x \leq b$. As funções $f_1, f_2, \dots, f_n$ são dadas e são funções das $n+1$ variáveis: a variável independente, x, e as n variáveis dependentes, $y_1(x), y_2(x), \dots, y_n(x)$.

Se o sistema tem solução, então haverá uma família de n soluções. Obtém-se uma única família de soluções se forem especificadas as n *condições auxiliares*. Tal como para uma única equação, se estas condições forem especificadas para o mesmo valor da variável independente, x, designadamente para o ponto inicial do intervalo

$$y_i(a) = c_i, \quad \text{para} \quad i = 1, 2, \dots, n,$$

o problema é designado por *sistema de equações diferenciais de primeira ordem com condições iniciais*.

9.3.2 Equações de Runge-Kutta e esquemas preditores-correctores

Qualquer dos métodos referidos até este momento, para a resolução numérica de uma equação diferencial de primeira ordem com condições iniciais, pode ser *estendido* e aplicado à resolução de um sistema de n equações diferenciais de primeira ordem com condições iniciais. A estratégia é a seguinte:

- Implementa-se a equação diferença do método a cada uma das equações do sistema, tendo em atenção a função f_i que define a equação i do sistema. A solução, que se define neste caso, como sendo uma família de valores

  $$y_{1,i}, y_{2,i}, y_{3,i}, \dots, y_{n,i},$$

  é assim gerada passo a passo, $i = 1, 2, \dots$, tal como se fazia no caso de uma equação.

A implementação das fórmulas numéricas não oferece qualquer dificuldade. Existem, no entanto, dois casos onde é preciso ter em atenção a sequência com que os cálculos são efectuados. São as fórmulas de Runge-Kutta e os esquemas preditores-correctores. Por exemplo,

- Na implementação do método de Runge-Kutta de segunda ordem, as equações que devem ser usadas para um sistema de n equações diferenciais são, para o passo i, $i = 0, 1 \ldots$

$$y_{1,i+1} = y_{1,i} + \frac{1}{2}(p_1 + q_1)$$

$$y_{2,i+1} = y_{2,i} + \frac{1}{2}(p_2 + q_2),$$

$$\ldots$$

e

$$y_{n,i+1} = y_{n,i} + \frac{1}{2}(p_n + q_n)$$

em que

$$p_1 = hf_1(x_i, y_{1,i}, y_{2,i}, \ldots, y_{n,i})$$

$$q_1 = hf_1(x_{i+1}, y_{1,i} + p_1, y_{2,i} + p_2, \ldots, y_{n,i} + p_n),$$

$$p_2 = hf_2(x_i, y_{1,i}, y_{2,i}, \ldots, y_{n,i})$$

$$q_2 = hf_2(x_{i+1}, y_{1,i} + p_1, y_{2,i} + p_2, \ldots, y_{n,i} + p_n),$$

$$\ldots$$

e

$$p_n = hf_n(x_i, y_{1,i}, y_{2,i}, \ldots, y_{n,i})$$

$$q_n = hf_n(x_{i+1}, y_{1,i} + p_1, y_{2,i} + p_2, \ldots, y_{n,i} + p_n).$$

Como, para determinar os q_1, q_2, ..., q_n, é preciso ter os valores de $p_1, p_2, \ldots, p_n$, os cálculos devem começar pelos valores de $p_1, p_2, \ldots, p_n$, seguidos dos $q_1, q_2, \ldots, q_n$ e finalmente, para cada equação, vêm os $y_{1,i+1}, y_{2,i+1}, \ldots, y_{n,i+1}$, referentes ao passo i.

- Na implementação do método preditor-corrector de Adams de segunda ordem, tem-se para cada equação do sistema, um conjunto de duas fórmulas

$$y^p_{1,i+2} = y_{1,i+1} + \frac{1}{2}h[3f_1(x_{i+1}, y_{1,i+1}, \ldots, y_{n,i+1}) - f_1(x_i, y_{1,i}, ..., y_{n,i})]$$

$$y^c_{1,i+2} = y_{1,i+1} + \frac{1}{2}h[f_1(x_{i+2}, y^p_{1,i+2}, \ldots, y^p_{n,i+2}) + f_1(x_{i+1}, y_{1,i+1}, \ldots, y_{n,i+1})],$$

$$y^p_{2,i+2} = y_{2,i+1} + \frac{1}{2}h[3f_2(x_{i+1}, y_{1,i+1}, \ldots, y_{n,i+1}) - f_2(x_i, y_{1,i}, \ldots, y_{n,i})]$$

$$y^c_{2,i+2} = y_{2,i+1} + \frac{1}{2}h[f_2(x_{i+2}, y^p_{1,i+2}, \ldots, y^p_{n,i+2}) + f_2(x_{i+1}, y_{1,i+1}, \ldots, y_{n,i+1})],$$

$$\ldots$$

e

$$y^p_{n,i+2} = y_{n,i+1} + \frac{1}{2}h[3f_n(x_{i+1}, y_{1,i+1}, \ldots, y_{n,i+1}) - f_n(x_i, y_{1,i}, ..., y_{n,i})]$$

$$y^c_{n,i+2} = y_{n,i+1} + \frac{1}{2}h[f_n(x_{i+2}, y^p_{1,i+2}, \ldots, y^p_{n,i+2}) + f_n(x_{i+1}, y_{1,i+1}, \ldots, y_{n,i+1})].$$

Cada valor corrigido é calculado em função dos valores previstos de todas as n variáveis dependentes, por isso, os cálculos devem ser iniciados pelos valores previstos $y^p_{1,i+2}, y^p_{2,i+2}, \ldots, y^p_{n,i+2}$ e só depois se calculam os valores corrigidos $y^c_{1,i+2}$, $y^c_{2,i+2}, \ldots,$ $y^c_{n,i+2}$.

O algoritmo 9.3 utiliza os dois métodos apresentados, equação de Runge-Kutta de segunda ordem e esquema preditor-corrector de Adams de segunda ordem, para a resolução de um sistema de n equações diferenciais de primeira ordem. O parâmetro *equa* faz a selecção entre as duas hipóteses.

Algoritmo 9.3 :

1. ler n, *equa* ("R.K." ou "P.C."), h, x_0 e *sup* e calcular *tol*
2. para $i = 1, \ldots, n$ ler $y_{i\,0}$
3. para $i = 1, \ldots, n$ introduzir as funções $f_i(x, y_1, y_2, \ldots, y_n)$
4. fazer $j = 0$
5. calcular $x_{j+1} = x_j + h$
6. se $(x_{j+1} - sup) \leq tol^{1/2}$ então
   - 6.1. para $i = 1, \ldots, n$ calcular $f_{i\,j} = f_i(x_j, y_{1\,j}, y_{2\,j}, \ldots, y_{n\,j})$, $p_i = h\, f_{i\,j}$ e $z_i = y_{i\,j} + p_i$,
   - 6.2. para $i = 1, \ldots, n$ calcular $g_i = f_i(x_{j+1}, z_1, z_2, \ldots, z_n)$ e $q_i = h\, g_i$
   - 6.3. para $i = 1, \ldots, n$ calcular $y_{i\,j+1} = y_{i\,j} + \frac{1}{2}(p_i + q_i)$
   - 6.4. se *equa* = "R.K."então fazer $j = j + 1$, $m = j$ e ir para o passo 5
   - 6.5. (senão)
     - 6.5.1. calcular $x_{j+2} = x_{j+1} + h$
     - 6.5.2. se $(x_{j+2} - sup) \leq tol^{1/2}$ então

6.5.2.1. para $i = 1, \ldots, n$ calcular $f_{i\,j+1} = f_i(x_{j+1}, y_{1\,j+1}, y_{2\,j+1}, \ldots, y_{n\,j+1})$, e $y_{i\,j+2} = y_{i\,j+1} + \frac{h}{2}(3f_{i\,j+1} - f_{i\,j})$

6.5.2.2. para $i = 1, \ldots, n$ calcular $g_i = f_i(x_{j+2}, y_{1\,j+2}, y_{2\,j+2}, \ldots, y_{n\,j+2})$

6.5.2.3. para $i = 1, \ldots, n$ calcular $y_{i\,j+2} = y_{i\,j+1} + \frac{h}{2}(g_i + f_{i\,j+1})$

6.5.2.4. fazer $j = j + 1$, $m = j + 1$ e ir para o passo 6.5.1

7. terminar com $y(x) \leftarrow ((y_{i\,j}, i = 1, \ldots, n), j = 0, 1 \ldots, m)$.

9.3.3 Implementação. Rotina SISDIF

A rotina SISDIF calcula a solução numérica de um sistema de equações diferenciais de primeira ordem, com condições iniciais e resulta da implementação do algoritmo 9.3. Pode-se escolher entre o método de Runge-Kutta ou o esquema preditor-corrector de Adams de ordem dois.

x_i	$y_{1,i}$	$y_{2,i}$	$y_{3,i}$	$y_{1,i}$	$y_{2,i}$	$y_{3,i}$
0.0	2.000000	0.000000	1.000000	2.000000	0.000000	1.000000
0.1	1.825000	0.255000	0.920000	-	-	-
0.2	1.687025	0.444975	0.868000	1.684038	0.450713	0.865250
0.3	1.577356	0.586506	0.836138	1.572801	0.595184	0.832014
0.4	1.489427	0.691947	0.818625	1.484277	0.701663	0.814060
0.5	1.418288	0.770501	0.811212	1.413118	0.780135	0.806748
0.6	1.360190	0.829023	0.810787	1.355320	0.837966	0.806714
0.7	1.312294	0.872622	0.815084	1.307879	0.880586	0.811535
0.8	1.272436	0.905104	0.822461	1.268532	0.911995	0.819472
0.9	1.238963	0.929302	0.831735	1.235569	0.935143	0.829288
1.0	1.210605	0.947330	0.842064	1.207687	0.952203	0.840110
...						
2.0	1.069298	0.997226	0.933476	1.068667	0.997741	0.933592
...						
3.0	1.025102	0.999854	0.975045	1.024871	0.999893	0.975236
...						
4.0	1.009228	0.999992	0.990780	1.009122	0.999995	0.990883
...						
5.0	1.003400	1.000000	0.996601	1.003351	1.000000	0.996649
...						
10.0	1.000023	1.000000	0.999977	1.000022	1.000000	0.999978

Exemplo 9.7 Considere o seguinte sistema de equações diferenciais lineares

$$\begin{cases} y_1'(x) = -y_1(x) + y_2(x) \\ y_2'(x) = y_1(x) - 2y_2(x) + y_3(x) \\ y_3'(x) = y_2(x) - y_3(x) \end{cases}$$

com $y_1(0) = 2$, $y_2(0) = 0$ e $y_3(0) = 1$. A solução numérica calculada para um passo de 0.1 é a apresentada na tabela anterior.

Os resultados das $2^{a}\cdot$, $3^{a}\cdot$ e $4^{a}\cdot$ colunas correspondem à fórmula de Runge-Kutta de segunda ordem. As restantes, $5^{a}\cdot$, $6^{a}\cdot$ e $7^{a}\cdot$ colunas apresentam os resultados obtidos pelo esquema preditor-corrector de Adams de segunda ordem; os valores auxiliares $y_{1,1}, y_{2,1}$ e $y_{3,1}$ foram calculados pela fórmula de Runge-Kutta da mesma ordem.

Quando o passo foi aumentado para $h = 1$, qualquer dos métodos induziu instabilidade. Alguns valores são apresentados na seguinte tabela:

x_i	$y_{1,i}$	$y_{2,i}$	$y_{3,i}$	$y_{1,i}$	$y_{2,i}$	$y_{3,i}$
0.0	2.000000	0.000000	1.000000	2.000000	0.000000	1.000000
1.0	2.500000	-1.500000	2.000000	-	-	-
2.0	4.250000	-5.250000	4.000000	5.875000	-8.625	5.75
...						
5.0	49.843750	-96.656250	49.812500	356.751953	-710.537109	356.785156
...						
10.0	4769.372070	-9535.743164	4769.371094	474075.111	-948147.222	474075.110

9.4 Equações diferenciais de ordem igual ou superior a dois

9.4.1 Introdução

A forma geral implícita de uma *equação diferencial ordinária de ordem* n, na variável dependente $y(x)$, sendo x a variável independente, é

$$F(x, y(x), y'(x), y''(x), y'''(x), \ldots, y^{(n-1)}(x), y^{(n)}(x)) = 0.$$

Esta equação pode ser resolvida explicitamente em ordem à derivada de $y(x)$, de ordem mais elevada, originando a equação

$$y^{(n)}(x) = f(x, y(x), y'(x), y''(x), ..., y^{(n-1)}(x)) \tag{9.34}$$

que com as n condições iniciais

$$y(a) = c_0,\ y'(a) = c_1,\ y''(a) = c_2,\ \ldots,\ y^{(n-1)}(a) = c_{n-1}$$

define um problema de condições iniciais, com solução única no intervalo $[a, b]$.

9.4.2 Resolução numérica

Para o caso geral de um problema com uma equação diferencial de ordem n com condições iniciais, apresentado na forma (9.34), a resolução numérica baseia-se na *transformação da equação diferencial explícita de ordem n, num sistema de n equações diferenciais de primeira ordem.* Para se atingir este objectivo, é necessário definir $n - 1$ novas variáveis dependentes. Designe-se $y(x)$ por $y_1(x)$,

$$y_1(x) \leftarrow y(x),$$

então, as novas variáveis dependentes $y_2(x), y_3(x), \ldots, y_n(x)$ são definidas da seguinte forma:

$$\begin{aligned} y_2(x) &= y_1'(x) \quad \text{ou} \quad y'(x) \\ y_3(x) &= y_2'(x) \quad \text{ou} \quad y''(x) \\ y_4(x) &= y_3'(x) \quad \text{ou} \quad y'''(x) \\ &\ldots \end{aligned}$$

e

$$y_n(x) = y_{n-1}' \quad \text{ou} \quad y^{(n-1)}(x).$$

As condições iniciais passam a ser

$$y_1(a) = c_0,\ y_2(a) = c_1,\ y_3(a) = c_2,\ \ldots,\ y_n(a) = c_{n-1}.$$

Da forma como foram definidas as novas variáveis, conclui-se que o sistema resultante, e equivalente, das n equações diferenciais de primeira ordem tem a forma

$$\begin{cases} y_1' = y_2 \\ y_2' = y_3 \\ y_3' = y_4 \\ \ldots \\ y_{n-1}' = y_n \\ y_n' = f(x, y_1, y_2, y_3, ..., y_{n-1}, y_n) \end{cases}$$

em que a função $f(x, y_1, \ldots, y_n)$ é a função que define a equação diferencial de ordem superior (veja-se (9.34)).

A resolução numérica deste sistema segue a estratégia apresentada em 9.3. Da primeira equação deste sistema tira-se que a função $f_1(x, y_1, y_2, \ldots, y_n)$ é definida por y_2, da segunda, conclui-se que $f_2(x, y_1, y_2, \ldots, y_n) = y_3$, e assim por adiante, até se chegar à última equação que apresenta a expressão mais complexa. Assim, $f_n(x, y_1, y_2, \ldots, y_n) = f(x, y_1, y_2, \ldots, y_n)$. Transformado o problema original (9.34) num sistema de equações diferenciais de primeira ordem, o algoritmo 9.3 é o mais indicado para a sua resolução.

O exemplo seguinte ilustra a estratégia apresentada.

Exemplo 9.8 Na resolução da seguinte equação diferencial de segunda ordem

$$(x+1)^2 y''(x) + (x+1)y'(x) + ((x+1)^2 - 0.25)y(x) = 0$$

com $y(0) = 0.6713967071418030\ldots$ e $y'(0) = 0.0954005144474446\ldots$, o sistema de equações de primeira ordem equivalente é o seguinte

$$\begin{cases} y_1'(x) = y_2(x) \\ y_2'(x) = -\frac{1}{x+1} y_2(x) - \frac{(x+1)^2 - 0.25}{(x+1)^2} y_1(x) \end{cases}$$

em que $y_1(0) = 0.6713967071418030\ldots$ e $y_2(0) = 0.0954005144474446\ldots$

Implementando o algoritmo 9.3 na resolução deste sistema (rotina SISDIF) para $0 \leq x \leq 1$, obtêm-se os resultados das colunas 2 e 3 da tabela, quando foi seleccionado o método de Runge-Kutta de segunda ordem com $h = 0.1$. Os resultados das duas colunas seguintes correspondem ao mesmo método, mas agora com $h = 0.25$.

x_i	$y_i = y_{1,i}$	$y_i' = y_{2,i}$	$y_i = y_{1,i}$	$y_i' = y_{2,i}$	$y_i = y_{1,i}$	$y_i' = y_{2,i}$	$y_i = y_{1,i}$	$y_i' = y_{2,i}$
0.0	0.671397	0.095401	0.671397	0.095401	0.671397	0.095401	0.671397	0.095401
0.1	0.677942	0.036827	-	-	...	...	-	-
0.2	0.678768	-0.019059	-	-	0.678837	-0.018995	-	-
0.25	-	-	0.676530	-0.047035	-	-	...	...
0.3	0.674137	-0.072318	-	-	0.674277	-0.072197	-	-
0.4	0.664311	-0.122914	-	-	0.664527	-0.122744	-	-
0.5	0.649561	-0.170745	0.648188	-0.172256	0.649860	-0.170538	0.649255	-0.170989
0.6	0.630168	-0.215675	-	-	0.630556	-0.215444	-	-
0.7	0.606432	-0.257545	-	-	0.606914	-0.257304	-	-
0.75	-	-	0.590707	-0.278846	-	-	0.593090	-0.276845
0.8	0.578665	-0.296190	-	-	0.579247	-0.295953	-	-
0.9	0.547199	-0.331448	-	-	0.547888	-0.331231	-	-
1.0	0.512380	-0.363164	0.509023	-0.364379	0.513168	-0.362981	0.512999	-0.362253

Seleccionando o esquema preditor-corrector de Adams de segunda ordem para a resolução do sistema e considerando o passo igual a 0.1 obtêm-se os resultados das colunas 6 e 7. As colunas 8 e 9 correspondem ao mesmo método anterior com $h = 0.25$.

9.5 Equações diferenciais 'stiff'

9.5.1 Introdução teórica

As equações diferenciais 'stiff' são equações cuja solução é formada por componentes que variam em proporções muito diferentes. Considere o exemplo:

Exemplo 9.9 A equação diferencial de primeira ordem

$$y'(x) = -100y(x) + 99e^{-x} \tag{9.35}$$

para $0 \leq x \leq 1$, com $y(0) = 0$ tem como solução analítica a função $y(x) = e^{-100x} + e^{-x}$. Para valores de x do intervalo $[0, 0.1]$, o termo e^{-x} toma valores entre 1 e ≈ 0.90, no entanto, e^{-100x} varia entre 1 e 0.000045.... A partir de $x = 0.1$ até $x = 1$, o termo e^{-100x} vai contribuindo cada vez menos para a solução. A resolução numérica da equação, no intervalo $[0, 0.1]$, para valores moderados de h, por exemplo $h = 0.1$, gera resultados disparatados. Torna-se necessário utilizar, nesta região, valores de h muito pequenos. Mas, logo que a componente da solução que varia mais lentamente começa a dominar, a estimativa do erro sugere um valor de h maior.

Quando se resolve uma equação diferencial 'stiff', a instabilidade causada por valores grandes de h origina também grandes erros locais. Um aumento brusco de erro local é indicativo de que a equação é 'stiff'. É preciso utilizar valores de h muito pequenos para que os erros locais não aumentem muito. Esta diminuição vai originar muitos passos e consequentemente aumenta o número de operações e o erro de arredondamento. Os valores da tabela 9.1 correspondem aos resultados obtidos pela aplicação dos métodos de Euler, Runge-Kutta de segunda ordem e Runge-Kutta de quarta ordem, respectivamente nas $2^{a}\cdot$, $3^{a}\cdot$ e $4^{a}\cdot$ colunas, para um passo de $h = 0.1$, na resolução do problema (9.35).

x_i	$y_i(E)$	$y_i(R-K(2))$	$y_i(R-K(4))$
0.0	0.0	0.0	0.0
0.1	9.9	-40.071	-289.993
0.2	-80.142	-1679.171	-84650.373
0.3	729.384	-68878.820	-24633496.08

Tabela 9.1: Resolução numérica do problema (9.35) com $h = 0.1$.

Mesmo para $h = 0.02$, os métodos de Euler e Runge-Kutta de segunda ordem induzem alguma instabilidade. Os valores obtidos para y, quando $x = 0.1$ e $x = 0.2$ são apresentados nas três primeiras colunas de valores de y_i, na tabela 9.2.

x_i	(E)	$(R-K(2))$	$(R-K(4))$	(E)	$(R-K(2))$	$(R-K(4))$	$y(x_i)$
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.1	1.904646	-0.094211	0.900768	0.904792	0.903907	0.904784	0.904792
0.2	-0.181251	-0.179457	0.818756	0.818690	0.818771	0.818732	0.818731

Tabela 9.2: Resolução numérica do problema (9.35) com $h = 0.02$ e $h = 0.01$.

Somente para $h = 0.01$ se obtêm valores aceitáveis, próximos da solução exacta. As quarta, quinta e sexta colunas de valores de y_i têm os resultados respeitantes, respectivamente, aos métodos de Euler e Runge-Kutta de segunda e quarta ordens. A última coluna apresenta os valores exactos de $y(x)$.

Existem métodos especiais para a resolução numérica deste tipo de equações. O método baseado na equação de Gear é um deles.

9.5.2 Fórmula de Gear

Existe um conjunto de fórmulas que resolve as equações 'stiff' satisfatoriamente e que são conhecidas por fórmulas de Gear[6]. A equação diferença obtém-se a partir de uma fórmula interpoladora baseada em diferenças ascendentes.

Para calcular o valor de $y(x)$, para um ponto $x = x_{i+3} + kh$, pode-se usar a fórmula interpoladora de Gregory-Newton baseada em diferenças ascendentes (veja-se em 6.2.3, do Capítulo 6),

$$y(x_{i+3}+kh) = y(x_{i+3}) + k\nabla y_{i+3} + \frac{k(k+1)}{2!}\nabla^2 y_{i+3} + \frac{k(k+1)(k+2)}{3!}\nabla^3 y_{i+3} + \cdots \quad (9.36)$$

em que k é uma constante negativa e h é o espaçamento constante entre os pontos, isto é, $x_{i+2} = x_{i+3} - h$, $x_{i+1} = x_{i+3} - 2h$, ... Se a fórmula for truncada após o quarto termo, do lado direito, em função da diferença ascendente de ordem três, a função $y(x)$ é aproximada por um polinómio de grau três,

$$y(x) \approx p_3(x) = y(x_{i+3}) + k\nabla y_{i+3} + \frac{k(k+1)}{2!}\nabla^2 y_{i+3} + \frac{k(k+1)(k+2)}{3!}\nabla^3 y_{i+3}. \quad (9.37)$$

[6]C.W. Gear é um matemático que nasceu em 1935 e estudou em Cambridge. É muito conhecido entre os engenheiros pelo seu livro publicado, em 1971, em que descreve e implementa uma classe de métodos para a resolução de equações 'stiff'. Como reconhecimento do seu trabalho, foi, em 1986, eleito Presidente da SIAM (Society for Industrial and Applied Mathematics) (Kahaner, Moler e Nash (1989)).

Como o objectivo é definir uma aproximação à derivada de $y(x)$, no ponto $x = x_{i+3}$, $y'(x_{i+3})$, começa-se por derivar o polinómio (9.37), em ordem a x, sabendo que $k = \frac{x-x_{i+3}}{h}$,

$$\frac{dp_3(x)}{dk}\frac{dk}{dx} = (\nabla y_{i+3} + \frac{2k+1}{2}\nabla^2 y_{i+3} + \frac{3k^2+6k+2}{6}\nabla^3 y_{i+3})\frac{1}{h}.$$

Fazendo $k \to 0$,

$$y'(x_{i+3}) \approx p_3'(x_{i+3}) = \frac{1}{h}(\nabla y_{i+3} + \frac{1}{2}\nabla^2 y_{i+3} + \frac{1}{3}\nabla^3 y_{i+3})$$

e usando as definições das diferenças,

$$\nabla y_{i+3} = y_{i+3} - y_{i+2},$$

$$\nabla^2 y_{i+3} = y_{i+3} - 2y_{i+2} + y_{i+1}$$

e

$$\nabla^3 y_{i+3} = y_{i+3} - 3y_{i+2} + 3y_{i+1} - y_i$$

obtém-se

$$y'(x_{i+3}) \approx y'_{i+3} = \frac{1}{6h}(11y_{i+3} - 18y_{i+2} + 9y_{i+1} - 2y_i).$$

A forma final da *fórmula de Gear* é, assim,

$$y_{i+3} = \frac{1}{11}(18y_{i+2} - 9y_{i+1} + 2y_i + 6hf(x_{i+3}, y_{i+3})) \tag{9.38}$$

para $i = 0, 1, \ldots$, é uma equação *implícita* e de *três passos*.

Esta característica exige o cálculo dos dois valores auxiliares y_1 e y_2, conhecido que é o valor inicial, $y(x_0)$.

O erro de truncatura cometido com a aproximação obtida pela equação (9.38) é dado por

$$e_T = -\frac{3}{2}h^4 y^{(iv)}(\xi_i) \quad \text{para } \xi_i \in [x_i, x_{i+3}]. \tag{9.39}$$

O cálculo dos valores auxiliares y_1 e y_2 deve ser feito através da implementação de uma fórmula de passo único. Podem ser usadas as fórmulas de Runge-Kutta de segunda ou de terceira ordem. É preciso algum cuidado com o cálculo destes valores auxiliares, uma vez que pode ser induzida instabilidade se o valor de h não for suficientemente pequeno. Valores de h menores ou iguais a 0.01 têm sido utilizados com algum sucesso. Para certos problemas é necessário baixar h para valores da ordem de 0.0001.

Como a fórmula de Gear é implícita, a sua implementação é idêntica à descrita em 9.2.5 (fórmula de Adams-Moulton) com o exemplo 9.3, se a equação (ou o sistema de equações) for linear em y. O algoritmo 9.4 implementa a fórmula de Gear e pode ser directamente aplicado a equações lineares.

No algoritmo, n indica o número de equações diferenciais no problema, $c_{ij}(x)$ representa o coeficiente da equação i referente à variável y_j e $termo_i(x)$ refere-se ao termo da equação i que não depende de y. Se a equação for não linear em y, há necessidade de introduzir alterações no passo 3, onde seriam introduzidas as funções genéricas f_i, $i = 1, \ldots, n$, e no 10, onde se resolveriam equações não lineares em $y_{ij+1},\ i = 1, \ldots, n$.

O parâmetro esc vai definir o factor de redução - $1/esc$ - do passo h para o cálculo dos valores auxiliares y_1 e y_2, pela fórmula de Runge-Kutta de segunda ordem, isto é: $h(R-K) = h(Gear)/esc$. Se for verificada instabilidade nos valores y_1 e y_2, o valor de esc deve ser aumentado.

Algoritmo 9.4 :

1. ler n, h, x_0, sup e esc e calcular tol
2. para $i = 1, \ldots, n$ ler $y_{i\,0}$ e fazer $r_{i\,0} = y_{i\,0}$
3. para $i = 1, \ldots, n$
   - 3.1. para $j = 1, \ldots, n$ introduzir as funções $c_{ij}(x)$
   - 3.2. introduzir as funções $termo_i(x)$
4. fazer $j = 0$, $t_0 = x_0$, $x_1 = x_0 + h$ e $x_2 = x_0 + 2h$
5. calcular $t_{j+1} = t_j + \frac{h}{esc}$
6. se $(t_{j+1} - x_2) \leq tol^{1/2}$ então
   - 6.1. para $i = 1, \ldots, n$ calcular $f_{i\,j} = f_i(t_j, r_{1\,j}, r_{2\,j}, \ldots, r_{n\,j})$, $p_i = \frac{h}{esc} f_{i\,j}$ e $z_i = r_{i\,j} + p_i$,
   - 6.2. para $i = 1, \ldots, n$ calcular $g_i = f_i(t_{j+1}, z_1, z_2, \ldots, z_n)$ e $q_i = \frac{h}{esc} g_i$
   - 6.3. para $i = 1, \ldots, n$ calcular $r_{i\,j+1} = r_{i\,j} + \frac{1}{2}(p_i + q_i)$
   - 6.4. fazer $j = j + 1$ e ir para o passo 5
7. fazer $j = 2$
8. para $i = 1, \ldots, n$ fazer $y_{i1} = r_{i\,esc}$ e $y_{i\,2} = r_{i\,2esc}$
9. calcular $x_{j+1} = x_j + h$
10. se $(x_{j+1} - sup) \leq tol^{1/2}$ então
    - 10.1. para $i = 1, \ldots, n$
      - 10.1.1. calcular $b_i = 6\,h\,termo_i(x_{j+1}) + 18y_{i\,j} - 9y_{i\,j-1} + 2y_{i\,j-2}$
      - 10.1.2. para $k = 1, \ldots, n$
        - 10.1.2.1. se $k = i$ então calcular $a_{i\,i} = 11 - 6\,h\,c_{i\,i}(x_{j+1})$
        - 10.1.2.2. senão calcular $a_{i\,k} = -6hc_{i\,k}(x_{j+1})$
    - 10.2. se $n = 1$ então $y_{1\,j+1} = \frac{b_1}{a_{1\,1}}$
    - 10.3. senão resolver o sistema de n equações $Ax = b$ (algoritmo 3.3) e colocar a solução em $y_{1\,j+1}, y_{2\,j+1}, \ldots, y_{n\,j+1}$
    - 10.4. fazer $j = j + 1$, $m = j$ e ir para o passo 9
11. terminar com $y(x) \leftarrow ((y_{i\,j}, i = 1, \ldots, n), j = 0, 1 \ldots, m)$.

9.5.3 Implementação. Rotina EQSTIF

A rotina EQSTIF implementa o algoritmo 9.4 e utiliza a fórmula de Gear para a resolução de equações diferenciais 'stiff' lineares.

Exemplo 9.10 Para resolver a equação

$$y'(x) = -100y(x) + 100x + 1$$

com $y(0) = 1$, para o intervalo $[0, 1]$, com um passo de $h = 0.1$, foi utilizada a fórmula de Runge-Kutta de segunda ordem, mas com um passo de 0.01, o que equivale a $esc = 10$ no algoritmo, para se calcularem os y_1 e y_2, aproximações a $y(x)$ para $x = 0.1$ e $x = 0.2$. A tabela seguinte apresenta alguns valores obtidos ao longo dos vinte passos:

x_i	0.0	0.01	0.02	...	0.1	...	0.2
$y_i(R - K(2))$	1.000000	0.510000	0.270000		0.100977		0.200001

A equação de Gear, usa $y_0 = 1$ (condição inicial), $y_1 = 0.100977$ e $y_2 = 0.200001$ (estes obtidos pela fórmula de Runge-Kutta) para calcular os restantes valores (agora com $h = 0.1$). Para este exemplo, tem-se $n = 1$, $c_{11}(x) = -100$ e $termo_1(x) = 100x + 1$.

x_i	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
y_i	0.328045	0.407138	0.498254	0.599443	0.700281	0.800093	0.899972	0.999989

A solução exacta desta equação é $y(x) = e^{-100x} + x$, donde se conclui que a contribuição do termo e^{-100x} só é significativa nas primeiras nove casas decimais, nos pontos $x = 0.1$, em que $y(0.1) = 0.100045399$ e $x = 0.2$, em que $y(0.2) = 0.200000002$.

Exemplo 9.11 Considere a equação diferencial de 2^a ordem

$$y''(x) = -101y'(x) - 100y(x)$$

em que $y(0) = 1.01$ e $y'(0) = -2$. Determinando o sistema de equações de primeira ordem equivalente

$$\begin{cases} y_1'(x) = y_2(x) \\ y_2'(x) = -100y_1(x) - 101y_2(x) \end{cases}$$

com $y_{10} = 1.01$ e $y_{20} = -2$, tem-se $c_{11}(x) = 0, c_{12}(x) = 1, c_{21}(x) = -100, c_{22}(x) = -101, termo_1(x) = 0$ e $termo_2(x) = 0$. A rotina EQSTIF fornece os seguintes resultados:

x_i	$y_i = y_{1i}$	$y_i' = y_{2i}$	$y_i = y_{1i}$	$y_i' = y_{2i}$
0.0	1.010000	-2.000000	1.010000	-2.000000
...	ao fim de 10	passos, $h = 0.01$:	ao fim de 5	passos, $h = 0.02$:
0.1	0.904849	-0.905815	0.914844	-1.904844
...	e ao fim de mais	10 passos	e ao fim de mais	5 passos
0.2	0.818734	-0.818734	0.828742	-1.818742
Os valores	obtidos pela	fórmula de	Gear com	$h = 0.1$:
0.3	0.741113	-0.768878	0.742391	-0.895771
0.4	0.670421	-0.677487	0.669765	-0.611045
0.5	0.606557	-0.604829	0.606518	-0.600076
0.6	0.548862	-0.548311	0.548977	-0.559108
0.7	0.496653	-0.496932	0.496674	-0.498405
0.8	0.449402	-0.449494	0.449396	-0.448369
0.9	0.406646	-0.406618	0.406649	-0.406455
1.0	0.367959	-0.367948	0.367965	-0.368095

As duas primeiras colunas identificadas com y_i e y_i' são obtidas quando se considera $esc = 10$, isto é, quando se utiliza a fórmula de Runge-Kutta com $h = 0.01$ para se calcularem os y_{11} e y_{12}, y_{21} e y_{22}. As duas colunas seguintes correspondem à utilização de Runge-Kutta com $h = 0.02$ (para $esc = 5$).

A solução exacta é a função $y(x) = 0.01e^{-100x} + e^{-x}$ e a contribuição do primeiro termo torna-se insignificante a partir de $x = 0.2$. Este valor é da ordem de 2.1×10^{-11} e $e^{-0.2} = 0.818730753$ (nenhum destes algarismos sofreria qualquer alteração com a soma do primeiro termo). Os resultados numéricos da 3^a coluna da tabela apresentam erros menores, especialmente no início da implementação, do que os da coluna 5. Mesmo assim o método de Gear consegue corrigir rapidamente os valores auxiliares não muito bons obtidos pela fórmula de Runge-Kutta (y_1' e y_2') quando $h = 0.02$. Veja a coluna 5 da tabela.

9.6 Problemas com condições de fronteira

9.6.1 Introdução teórica

Considere a equação diferencial ordinária de segunda ordem

$$\frac{d^2y(x)}{dx^2} = -cx^2$$

em que c é uma constante dada. Para calcular uma solução, no intervalo $0 < x < l$, é preciso especificar duas condições auxiliares. Se estas condições forem especificadas para diferentes valores da variável independente, x, nomeadamente nas fronteiras do domínio, o problema é identificado como problema de equações diferenciais ordinárias com *condições de fronteira*.

A resolução numérica deste tipo de problema de equações diferenciais pode ser feita através dos seguintes tipos de métodos:

i) métodos 'shooting', que são baseados na transformação do problema original num outro, equivalente, mas com condições iniciais. A resolução é então feita através dos métodos já referidos anteriormente, obtendo-se a solução numérica passo a passo. Este tipo de método não é muito aconselhável devido à falta de informação relativa a valores de y, ou das derivadas, no princípio do intervalo. Esta falha é ultrapassada fornecendo aproximações a estes valores e calculando a solução numérica, passo a passo. Como, na generalidade, estas aproximações não são muito boas, o valor calculado na fronteira superior do intervalo não coincide com a condição dada. O processo terá de ser repetido, começando-se com outros valores.

ii) Os *métodos baseados nas diferenças finitas* calculam a solução *toda de uma vez*, em todo o intervalo, calculando aproximações a $y(x)$ num conjunto de pontos desse intervalo. Se os pontos de $[a, b]$,

$$x_0, x_1, x_2, ..., x_{n-1}, x_n$$

estiverem igualmente espaçados de h, tem-se

$$a = x_0 < x_1 < x_2 < \ldots < x_{n-2} < x_{n-1} < x_n = b.$$

Aos pontos x_0 e x_n chamam-se *pontos fronteira* e aos $x_1, x_2 \ldots, x_{n-1}$ chamam-se *pontos interiores*.

9.6.2 Fórmulas baseadas em diferenças finitas

O método das diferenças finitas tem por base a substituição das derivadas existentes, quer na equação diferencial quer nas condições de fronteira, por aproximações baseadas nas diferenças finitas. São diferenças finitas as diferenças divididas, as centrais, as descendentes e as ascendentes (recordar o que foi dado no Capítulo 6).

As equações resultantes passam então a ser equações algébricas em y. São lineares, se a equação diferencial também o for. Neste caso, a resolução do sistema de equações algébricas resultante utiliza os métodos indicados no Capítulo 3.

Passamos, em seguida, à dedução de algumas fórmulas usadas para a aproximação das derivadas de várias ordens.

A partir das expansões em série de Taylor de $y(x)$, nos pontos $x_{i+1}, x_{i-1}, x_{i+2}$ e x_{i-2},

$$y(x_{i+1}) = y(x_i) + hy'(x_i) + \frac{1}{2}h^2y''(x_i) + \frac{1}{3!}h^3y'''(x_i) + \frac{1}{4!}h^4y^{(iv)}(x_i) + \\ + \frac{1}{5!}h^5y^{(v)}(x_i) + \mathcal{O}(h^6) \tag{9.40}$$

$$y(x_{i-1}) = y(x_i) - hy'(x_i) + \frac{1}{2}h^2y''(x_i) - \frac{1}{3!}h^3y'''(x_i) + \frac{1}{4!}h^4y^{(iv)}(x_i) - \\ - \frac{1}{5!}h^5y^{(v)}(x_i) + \mathcal{O}(h^6) \tag{9.41}$$

$$y(x_{i+2}) = y(x_i) + 2hy'(x_i) + \frac{1}{2}(2h)^2y''(x_i) + \frac{1}{3!}(2h)^3y'''(x_i) + \\ + \frac{1}{4!}(2h)^4y^{(iv)}(x_i) + \frac{1}{5!}(2h)^5y^{(v)}(x_i) + \mathcal{O}(h^6) \tag{9.42}$$

e

$$y(x_{i-2}) = y(x_i) - 2hy'(x_i) + \frac{1}{2}(2h)^2y''(x_i) - \frac{1}{3!}(2h)^3y'''(x_i) + \\ + \frac{1}{4!}(2h)^4y^{(iv)}(x_i) - \frac{1}{5!}(2h)^5y^{(v)}(x_i) + \mathcal{O}(h^6), \tag{9.43}$$

podem-se deduzir fórmulas para aproximar as primeira, segunda, terceira e quarta derivadas. Subtraindo (9.41) de (9.40), obtém-se

$$y(x_{i+1}) - y(x_{i-1}) = 2hy'(x_i) + \frac{2}{3!}h^3y'''(x_i) + \mathcal{O}(h^4)$$

ou seja

$$y'(x_i) \approx \frac{1}{2h}[y(x_{i+1}) - y(x_{i-1})] \tag{9.44}$$

com erro de truncatura $\mathcal{O}(h^2)$. Adicionando as equações (9.40) e (9.41) tem-se

$$y(x_{i+1}) + y(x_{i-1}) = 2y(x_i) + h^2 y''(x_i) + \frac{2}{4!}h^4 y^{(iv)}(x_i) + \mathcal{O}(h^6)$$

ou

$$y''(x_i) \approx \frac{1}{h^2}[y(x_{i+1}) - 2y(x_i) + y(x_{i-1})] \tag{9.45}$$

com erro de truncatura $\mathcal{O}(h^2)$. Combinando as quatro equações da seguinte maneira: (9.42) - 2 $\times$(9.40) + 2 $\times$(9.41) - (9.43) consegue-se

$$y(x_{i+2}) - 2y(x_{i+1}) + 2y(x_{i-1}) - y(x_{i-2}) = 2h^3 y'''(x_i) + \mathcal{O}(h^5)$$

e a aproximação à terceira derivada é,

$$y'''(x_i) \approx \frac{1}{2h^3}[y(x_{i+2}) - 2y(x_{i+1}) + 2y(x_{i-1}) - y(x_{i-2})] \tag{9.46}$$

cujo erro de truncatura é $\mathcal{O}(h^2)$. Finalmente, para se conseguir uma aproximação à quarta derivada, combinam-se as quatro equações da seguinte forma: (9.42) - 4 $\times$(9.40) - 4 $\times$(9.41) + (9.43). De

$$y(x_{i+2}) - 4y(x_{i+1}) - 4y(x_{i-1}) + y(x_{i-2}) = -6y(x_i) + h^4 y^{(iv)}(x_i) + \mathcal{O}(h^6)$$

tira-se que

$$y^{(iv)}(x_i) \approx \frac{1}{h^4}[y(x_{i+2}) - 4y(x_{i+1}) + 6y(x_i) - 4y(x_{i-1}) + y(x_{i-2})] \tag{9.47}$$

e o erro de truncatura é, mais uma vez, $\mathcal{O}(h^2)$.

Das fórmulas interpoladoras com diferenças centrais, designadamente da fórmula de Stirling

$$y(x_i + kh) = y(x_i) + k\mu\delta y(x_i) + \frac{k}{2}k\delta^2 y(x_i) + \cdots$$

derivando em ordem a k

$$y'(x_i + kh)h = \mu\delta y(x_i) + k\delta^2 y(x_i) + \frac{3k^2 - 1}{6}\mu\delta^3 y(x_i) + \cdots \tag{9.48}$$

e fazendo $k \to 0$, obtém-se uma fórmula para aproximar a primeira derivada,

$$y'(x_i) = \frac{1}{h}(\mu\delta y(x_i) + \cdots)$$

ou

$$\begin{aligned} y'(x_i) &\approx \frac{1}{h}[\frac{1}{2}(\delta y(x_{i+\frac{1}{2}}) + \delta y(x_{i-\frac{1}{2}}))] \\ &\approx \frac{1}{h}[\frac{1}{2}((y(x_{i+1}) - y(x_i)) + (y(x_i) - y(x_{i-1}))] \\ &\approx \frac{1}{2h}[y(x_{i+1}) - y(x_{i-1})] \end{aligned}$$

com erro de truncatura $\mathcal{O}(h^2)$. Da mesma fórmula, se deduz uma aproximação para a segunda derivada. Pegando na expressão (9.48) e derivando novamente em ordem a k, tem-se

$$y''(x_i + kh)h^2 = \delta^2 y(x_i) + k\mu\delta^3 y(x_i) + \cdots$$

Fazendo $k \to 0$, tem-se finalmente

$$y''(x_i) = \frac{1}{h^2}(\delta^2 y(x_i) + \cdots)$$

ou

$$y''(x_i) \approx \frac{1}{h^2}[y(x_{i+1}) - 2y(x_i) + y(x_{i-1})] \tag{9.49}$$

com erro de truncatura da ordem de h^2. Esta fórmula é igual à deduzida em (9.45).

A partir da fórmula interpoladora de Gregory-Newton com diferenças ascendentes (veja-se a equação (6.6) de 6.2.3 do Capítulo 6) podem-se deduzir outras fórmulas de aproximação às derivadas. Os passos da dedução são idênticos aos usados nos dois casos anteriores. Estas fórmulas são utilizadas para aproximar as derivadas que se encontram nas condições de fronteira, referentes ao *ponto da fronteira superior do intervalo*. Tem-se, assim, para aproximar a primeira, segunda e terceira derivadas,

$$y'(x_i) \approx \frac{1}{h}[y(x_i) - y(x_{i-1})], \tag{9.50}$$

$$y''(x_i) \approx \frac{1}{h^2}[y(x_i) - 2y(x_{i-1}) + y(x_{i-2})] \tag{9.51}$$

e

$$y'''(x_i) \approx \frac{1}{h^3}[y(x_i) - 3y(x_{i-1}) + 3y(x_{i-2}) - y(x_{i-3})]. \tag{9.52}$$

As fórmulas de aproximação das derivadas, deduzidas a partir da fórmula interpoladora de Gregory-Newton com diferenças descendentes (equação (6.3) de 6.2.2 do Capítulo 6), são aconselháveis na substituição das derivadas que surgem nas condições de fronteira referentes ao *ponto da fronteira inferior do intervalo*. Três dos casos mais utilizados são:

$$y'(x_i) \approx \frac{1}{h}[y(x_{i+1}) - y(x_i)], \tag{9.53}$$

$$y''(x_i) \approx \frac{1}{h^2}[y(x_{i+2}) - 2y(x_{i+1}) + y(x_i)] \tag{9.54}$$

e

$$y'''(x_i) \approx \frac{1}{h^3}[y(x_{i+3}) - 3y(x_{i+2}) + 3y(x_{i+1}) - y(x_i)]. \tag{9.55}$$

Todas as fórmulas de aproximação das derivadas tomam como ponto de referência o ponto x_i do intervalo. Depois de substituídas as derivadas da equação diferencial pelas fórmulas de aproximação, obtém-se, como já foi referido, uma equação algébrica em função de valores de $y(x)$ em diversos pontos do intervalo. Como existe *uma equação algébrica para cada ponto x_i do interior do intervalo*, $x_1, x_2, \ldots, x_{n-2}, x_{n-1}$, ao todo $n-1$ pontos, após a substituição, fica-se com um sistema de $n-1$ equações nas $n+1$ incógnitas $y_0, y_1, y_2, \ldots, y_{n-2}, y_{n-1}$ e y_n. Existem, nesta altura, mais incógnitas do que equações.

No entanto, nos exemplos mais simples de equações de segunda ordem, os valores de y_0 e y_n são dados pelas seguintes condições de fronteira,

$$y(x_0) = c_0 \quad \text{e} \quad y(x_n) = c_n$$

e o sistema passa a ter apenas $n-1$ incógnitas, pois

$$y_0 = c_0 \quad \text{e} \quad y_n = c_n.$$

Se as condições de fronteira envolverem derivadas de $y(x)$, a sua resolução já não é tão simples como o caso anterior. De acordo com o que foi referido atrás, as derivadas devem ser substituídas pelas fórmulas de aproximação (em geral usam-se as fórmulas (9.50), (9.51), (9.52), (9.53), (9.54) e (9.55)). As equações algébricas resultantes devem ser adicionadas às outras para formarem um sistema de $n+1$ equações para se calcularem os $y_0, y_1, \ldots, y_{n-1}, y_n$.

Os sistemas resultantes desta substituição, quando lineares, têm uma forma especial. A matriz dos coeficientes é uma matriz em banda. Nas equações de segunda ordem os elementos diferentes de zero aparecem sobre três diagonais, a diagonal principal, a diagonal acima da diagonal principal e a diagonal abaixo da diagonal principal, definindo uma matriz tridiagonal. Noutros problemas de ordem mais elevada, pode obter-se uma matriz em banda com cinco diagonais. Esta particularidade depende da ordem das derivadas presentes na equação diferencial.

Sendo difícil desenvolver um algoritmo geral, que pudesse ser aplicado a qualquer tipo de equação diferencial com condições de fronteira, optou-se por apresentar um que possa, pelo menos, ser utilizado para a resolução de uma equação diferencial de segunda ordem, *linear em y*, com coeficientes não constantes e que tenha como condições de fronteira *funções lineares* que dependam apenas de $y(x)$ e $y'(x)$. Esta opção define o algoritmo 9.5. Nos passos 2 e 3 do algoritmo são introduzidos os coeficientes. No passo 2, $c_2(x)$ é o coeficiente de $y''(x)$, $c_1(x)$ é o coeficiente de $y'(x)$, $c_0(x)$ o de $y(x)$ e $termo(x)$ o termo independente da equação, quando colocado no membro do lado direito. Da mesma forma,

os parâmetros que surgem no passo 3 do algoritmo correspondem às seguintes equações, na forma geral:

- da fronteira inferior

$$cinf_1(x)y'(x) + cinf_0(x)y(x) = tinf(x)$$

- da fronteira superior

$$csup_1(x)y'(x) + csup_0(x)y(x) = tsup(x).$$

Algoritmo 9.5 :

1. ler x_0 e sup
2. introduzir as funções $c_0(x)$, $c_1(x)$, $c_2(x)$ e $termo(x)$
3. introduzir as funções $cinf_0(x)$, $cinf_1(x)$, $tinf(x)$, $csup_0(x)$, $csup_1(x)$ e $tsup(x)$
4. ler h
5. calcular $n = \frac{sup - x_0}{h}$
6. se $n \neq$ inteiro ir para o passo 4
7. para $j = 1, \ldots, n-1$
   - 7.1. calcular $x_j = x_{j-1} + h$
   - 7.2. calcular $d_{j+1} = -\frac{2\,c_2(x_j)}{h^2} + c_0(x_j)$, $b_{j+1} = termo(x_j)$, $s_{j+1} = \frac{c_2(x_j)}{h^2} + \frac{c_1(x_j)}{2h}$ e $i_{j+1} = \frac{c_2(x_j)}{h^2} - \frac{c_1(x_j)}{2h}$
8. calcular $d_1 = -\frac{cinf_1(x_0)}{h} + cinf_0(x_0)$, $d_{n+1} = \frac{csup_1(sup)}{h} + csup_0(sup)$, $b_1 = tinf(x_0)$, $b_{n+1} = tsup(sup)$, $s_1 = \frac{cinf_1(x_0)}{h}$ e $i_{n+1} = -\frac{csup_1(x_n)}{h}$
9. resolver o sistema tridiagonal de $n+1$ equações, $Tx = b$ (algoritmo 3.4) e colocar a solução em $y_0, y_1, y_2, \ldots, y_{n-1}, y_n$
10. terminar com $y(x) \leftarrow (y_j, j = 0, 1, \ldots, n)$.

9.6.3 Implementação. Rotina EQUFRO

A implementação do algoritmo 9.5 origina a rotina EQUFRO. Os dois exemplos seguintes consideram dois casos que podem ser resolvidos pela rotina. No primeiro, as condições de fronteira não envolvem derivadas e no segundo, uma das condições envolve a primeira derivada.

Exemplo 9.12 Dada a equação diferencial de 2^a ordem

$$y''(x) + \frac{1}{x}y'(x) = -2$$

em que $y(0.2) = 0$ e $y(0.5) = 0$, para calcular a sua solução numérica no intervalo $[0.2, 0.5]$, considera-se $c_2(x) = 1, c_1(x) = \frac{1}{x}, c_0(x) = 0, termo(x) = -2, cinf_1(x) = 0, cinf_0(x) = 1, tinf(x) = 0, csup_1(x) = 0, csup_0(x) = 1$ e $tsup(x) = 0$:

- Para $h = 0.1$, passamos a ter dois pontos interiores, $x_1 = 0.3$ e $x_2 = 0.4$. A matriz dos coeficientes do sistema linear que resulta da substituição das derivadas pelas fórmulas de aproximação baseadas nas diferenças finitas é

$$\begin{pmatrix} 1 & 0 & & \\ 83.333333 & -200 & 116.666667 & \\ & 87.5 & -200 & 112.5 \\ & & 0 & 1 \end{pmatrix}$$

e o vector do lado direito fica $b = (0, -2, -2, 0)^T$. A solução é pois

$$(y_0, y_1, y_2, y_3) = (0, 0.021259, 0.019301, 0);$$

- Para $h = 0.05$, fica-se com 5 pontos interiores, $x_1 = 0.25, x_2 = 0.3, x_3 = 0.35, x_4 = 0.4$ e $x_5 = 0.45$. O sistema resultante da discretização é

$$\begin{pmatrix} 1 & 0 & & & & & & \\ 360 & -800 & 440 & & & & & \\ & 366.6666667 & -800 & 433.333333 & & & & \\ & & 371.428571 & -800 & 428.571429 & & & \\ & & & 375 & -800 & 425 & & \\ & & & & 377.777778 & -800 & 422.222222 & \\ & & & & & 0 & 1 & \end{pmatrix} y = \begin{pmatrix} 0 \\ -2 \\ -2 \\ -2 \\ -2 \\ -2 \\ 0 \end{pmatrix}$$

e a solução calculada é

$$(y_0, y_1, y_2, y_3, y_4, y_5, y_6) = (0, 0.014275, 0.021409, 0.022831, 0.019396, 0.011659, 0).$$

Exemplo 9.13 Para calcular a solução numérica da equação diferencial de segunda ordem

$$-2y''(x) + y(x) = e^{-0.2x}$$

em que $y(0) = 1$ e $y'(10) + y(10) = 0$, no intervalo $0 \leq x \leq 10$, com espaçamento entre pontos, $h = 1$, obtém-se para a primeira equação do sistema $y_0 = 1$ e para a última equação $-y_9 + 2y_{10} = 0$. As outras nove equações são

$$\begin{cases} -2y_0 & +5y_1 - 2y_2 & & = & e^{-0.2} \\ & -2y_1 + 5y_2 - 2y_3 & & = & e^{-0.4} \\ & \cdots & & & \\ & & -2y_8 + 5y_9 - 2y_{10} & = & e^{-1.8} \end{cases}$$

A solução do sistema de 11 equações é o vector

$$(1, 0.846435, 0.706723, 0.585211, 0.481900, 0.394873, 0.321344, 0.257889, 0.200081, 0.141365, 0.070683)^T.$$

Considerações sobre a dimensão e resolução numérica do sistema de equações algébricas:

i) Se o espaçamento entre os pontos, h, é uma quantidade muito pequena, existem muitos pontos no intervalo $[x_0, x_n]$ e o sistema algébrico resultante tem muitas equações e, consequentemente, muitas incógnitas, $y_0, y_1, y_2, \ldots, y_{n-1}, y_n$. O sistema é conhecido por sistema de *grande dimensão* com a maior parte dos elementos da matriz dos coeficientes igual a zero, sendo aconselhável, a implementação de métodos iterativos na sua resolução.

ii) Se o espaçamento entre os pontos, h, é uma quantidade razoável, o número de pontos no intervalo é pequeno e o sistema de equações algébricas é de pequena dimensão. Um método directo e estável é o mais aconselhável para a sua resolução. Se o sistema tem a forma tridiagonal, existem até algoritmos especiais para a sua resolução.

Considerações sobre a precisão dos resultados obtidos:

i) Se os erros de truncatura das fórmulas usadas para aproximar as derivadas forem da ordem de h^2, a solução numérica $y_0, y_1, y_2, ..., y_{n-1}, y_n$, aproximação à função $y(x)$, nos pontos $x_0, x_1, x_2, ..., x_{n-1}, x_n$, é também da mesma ordem de precisão, $\mathcal{O}(h^2)$.

ii) Quando se pretende diminuir o erro da solução numérica, deve-se diminuir o valor de h. Assim, se o novo valor de h for metade do valor antigo, uma aproximação de $\mathcal{O}(h^2)$ originará uma melhoria e o erro passará a ser um quarto do anterior.

9.7 Problemas

1. Dada a equação diferencial de primeira ordem

$$y'(x) = cos(x)y(x)$$

com $y(0) = 1$, determine a solução numérica no intervalo $[0, 2]$ considerando $h = 0.5$ e usando

   a) o método implícito de Adams-Moulton;

   b) o método de Runge-Kutta de segunda ordem;

   c) o método de Euler melhorado.

   Calcule os erros locais de truncatura, em cada passo, para as fórmulas de Adams-Moulton e Euler melhorado. Qual das fórmulas lhe parece que apresenta melhores resultados? Justifique.

2. Considere a seguinte equação diferencial

$$y'(x) = x^2 - y^2(x), \quad \text{com } y(0) = 1$$

   e $0 \leq x \leq 1$.

a) Para $h = 0.2$, use um método implícito de passo único e de segunda ordem para calcular numericamente $y(0.2)$.

b) Calcule um limite superior do erro de truncatura cometido neste passo, com a fórmula usada.

3. Considere a seguinte equação diferencial

$$y'(x) = -x\,(y(x))^2$$

com $y(0) = 2$.

a) Calcule a solução numérica, usando um método implícito de segunda ordem, no intervalo $0 \leq x \leq 2$ e considerando um espaçamento, h, constante e igual a 1.

b) Estime o erro de truncatura cometido na alínea anterior, no intervalo $[0, 1]$.

c) Usando o mesmo espaçamento, $h = 1$, implemente o método de Runge-Kutta de segunda ordem, para obter a solução no intervalo $[0, 2]$.

d) Estime o erro local de truncatura cometido, no primeiro passo, com o método de c).

4. Dada a equação diferencial de primeira ordem

$$y'(x) + \frac{1}{(1 + y(x)^2)} = 0$$

em que $y(0) = 1$, determine a solução numérica no intervalo $0 \leq x \leq 1$, considerando

a) o passo igual a 1 e a fórmula de Runge-Kutta de quarta ordem;

b) o passo igual a 0.1 e a mesma fórmula de a);

c) o passo igual a 0.1 e o esquema preditor-corrector de Adams de quarta ordem. Verifique que os valores obtidos neste caso coincidem com os valores obtidos em b) nas quatro primeiras casas decimais.

5. Dado o problema de equações diferenciais com condições iniciais

$$y'(x) + y(x)|y(x)| = 0$$

com $y(0) = 1$, determine a solução numérica no intervalo $0 \leq x \leq 5$, usando um passo $h = 0.5$ e uma fórmula de passo único.

6. Considere o seguinte sistema de equações diferenciais

$$\begin{cases} y_1'(x) = (x-1)y_1(x) - y_2(x) \\ y_2'(x) = y_3(x) - x^2 - y_1(x) \\ y_3'(x) = -2y_2(x) - y_3(x) \end{cases}$$

com $y_1(1) = 1$ e $y_2(1) = y_3(1) = -1$. Calcule a solução numérica, usando um método implícito e de passo único, no intervalo $1 \leq x \leq 1.5$ e considerando um espaçamento constante e igual a $h = 0.5$.

7. Considere o seguinte sistema de equações diferenciais não lineares

$$\begin{cases} y_1'(x) = y_2(x)y_3(x) \\ y_2'(x) = -y_1(x)y_3(x) \\ y_3'(x) = -0.51y_1(x)y_2(x) \end{cases}$$

em que $y_1(0) = 0$ e $y_2(0) = y_3(0) = 1$. Calcule a solução numérica no intervalo $[0, 10]$ considerando $h = 0.1$ e usando o método de Runge-Kutta de segunda ordem. Repita, mas agora, com $h = 1$. Que conclusões pode tirar desta segunda implementação?

8. Pretende-se resolver numericamente a seguinte equação diferencial

$$\frac{d^2\theta}{dt^2} = -b^2 \frac{S}{l} sen(\theta), \quad \text{com } \theta(0) = 1 \text{ e } \frac{d\theta}{dt}(0) = 0$$

para $0 \leq t \leq 1$.

Considere $b = 0.5$, $S = 2$ e $l = 1$. Tome $h = 0.5$ e use um método de passo único e de 2^a ordem.

9. A clássica lei do inverso do quadrado para um objecto que cai para uma massa atractiva, por exemplo a Terra, é

$$x''(t) = -\frac{g\,R^2}{(x(t))^2}$$

em que g é uma constante ($= 6.0606 \times 10^{-3}$) e R é o raio da Terra ($= 4000$). Se a altitude inicial do objecto for de 200 e a velocidade inicial for nula, *i.e.*,

$$x(0) = 200 \quad \text{e} \quad x'(0) = 0,$$

calcule a altitude do objecto ao fim de $t = 5$ segundos.

Nota: Considere um espaçamento h igual a 5 segundos.

O que terá acontecido ao objecto entre os 5 e os 10 segundos ?

Use um método de segunda ordem e de passo único.

10. Dado o problema de equações diferenciais

$$y'''(x) = x\,y(x) \quad \text{com } y(0) = 1\,, \; y'(0) = 0 \text{ e } y''(0) = 1$$

calcule aproximações numéricas a $y(0.4), y'(0.4)$ e $y''(0.4)$. Implemente o esquema preditor-corrector de Adams de segunda ordem e use $h = 0.2$.

Use, nos cálculos, as seguintes aproximações:

$$y(0.2) \approx 1.02\,, \; y'(0.2) \approx 0.2 \text{ e } y''(0.2) \approx 1.02$$

obtidas pela fórmula de Runge-Kutta de segunda ordem.

11. Resolva a equação diferencial de ordem três:

$$y'''(x) + 0.5y(x)y''(x) = 0$$

com $y(0) = y'(0) = 0$ e $y''(0) = 1$, no intervalo $[0, 1]$. Use um espaçamento, h, constante igual a 0.5.

12. Resolva o sistema de equações diferenciais

$$\begin{cases} y_1''(x) = 1 - 0.1y_1'(x) - y_1 + 0.1y_2'(x) + y_2(x) \\ y_2''(x) = 0.1y_1'(x) + y_1(x) - 0.1y_2'(x) - 2y_2(x) + y_3(x) \\ y_3''(x) = y_2(x) - 0.1y_3'(x) - 2y_3(x) \end{cases}$$

no intervalo $[0, 1]$, sabendo que $y_1(0) = y_1'(0) = y_2(0) = y_2'(0) = y_3(0) = y_3'(0) = 0$.

13. Resolva o seguinte sistema de equações de 1^a ordem

$$\begin{cases} y_1'(x) = -10y_1(x) + 6y_2(x) \\ y_2'(x) = 13.5y_1(x) - 10y_2(x) \end{cases}$$

no intervalo $[0, 1]$, sabendo que $y_{10} = 1.3333333\ldots$ e $y_{20} = 0$,

a) Usando o método de Runge-Kutta de 2^a ordem e um passo de 0.1.

b) Comente os resultados obtidos em a). Qual o método mais apropriado para resolver o sistema?

c) Use a sugestão indicada em b), para um passo de 0.1, para calcular a solução numérica do sistema.

14. Resolva o seguinte sistema de equações diferenciais 'stiff'

$$\begin{cases} y_1'(x) = y_2(x) \\ y_2'(x) = -1001y_1(x) - 1000y_2(x) \end{cases}$$

no intervalo $[0, 0.5]$, usando $h = 0.05$. As condições iniciais são $y_1(0) = 1$ e $y_2(0) = -1$.

15. Para $0 \leq x \leq 0.2$ e $k = 30$, calcule a solução da equação

$$-y''(x) = q(x)\frac{1}{k}$$

considerando

i) $q(x) = 200$

ii) $q(x) = 100e^{-10x}$

As condições são $y(0) = 0.1333333333...$, para o caso i) e $y(0) = 0.037466064$ para o caso ii), e $y(0.2) = 0$ para os dois casos.

16. Resolva a equação diferencial de 2^a ordem

$$y''(x) + \frac{2(y'(x))^2}{y(x)} - \frac{1}{2}y(x) = 0$$

no intervalo $[0, 1]$, sendo $y(0) = 1$ e $y(1) = 1.5$.

17. Dado o problema de equações diferenciais

$$(1 - x)y''(x) + x^2y'(x) - y(x) = x$$

com $y'(0) - y(0) = 1$ e $y(1) = 1$, calcule aproximações numéricas a $y(0), y(0.25), y(0.5)$ e $y(0.75)$.

18. Resolva o problema de equações diferenciais de segunda ordem

$$y''(x) + 5y'(x) + 6y(x) = 3e^{-x}$$

no intervalo $[0, 1]$, sendo $y(0) = 4$ e $y'(1) = -2$. Tome para o passo h o valor 0.05

19. Dado o problema de equações diferenciais

$$-\frac{d}{dx}(x\,\frac{dy}{dx}) + (1 + x)^2\, y = 2 - e^{-x} \quad \text{com } y(0) = 5 \text{ e } y(1) = -3$$

para $0 \leq x \leq 1$, calcule aproximações numéricas a $y(0.25), y(0.5)$ e $y(0.75)$.

Capítulo 10

Optimização não linear sem restrições

10.1 Introdução

10.1.1 Forma geral do problema

A formulação matemática de um problema de optimização, na sua forma mais geral, é

$$\text{minimizar}_{x \in \mathbf{R}^n} f(x) \tag{10.1}$$

sujeito a

$$c_i(x) = 0, \quad i = 1, \ldots, m$$

$$c_j(x) \geq 0, \quad j = m+1, \ldots, t$$

À função f que se pretende minimizar chama-se *função objectivo*. As n variáveis do problema $x_1, x_2, \ldots, x_{n-1}, x_n$ são manipuladas como um vector x, de n elementos ($x \in \mathbf{R}^n$). Os valores óptimos calculados pelo processo são designados por x^* e o correspondente valor de f é f^*.

As equações $c_i(x) = 0$, $i = 1, \ldots, m$ definem as restrições nas variáveis do tipo *igualdade* e as inequações $c_j(x) \geq 0$, $j = m+1, \ldots, t$ definem restrições nas variáveis do tipo *desigualdade*. Podem existir problemas onde só existam restrições de igualdade, outros onde só existam desigualdades e ainda outros sem qualquer tipo de restrição nas variáveis.

O problema de optimização formulado em (10.1) define um problema de *minimização*. Esta formulação não se torna restritiva uma vez que qualquer problema pode ser colocado nesta forma. Em particular, um problema de *maximização* pode ser resolvido implementando um método que calcule mínimos de funções, pois

$$\text{máximo } f(x) = -\text{mínimo } [-f(x)]$$

e o valor de x, onde f atinge o seu máximo é o mesmo onde $-f$ atinge o seu mínimo.

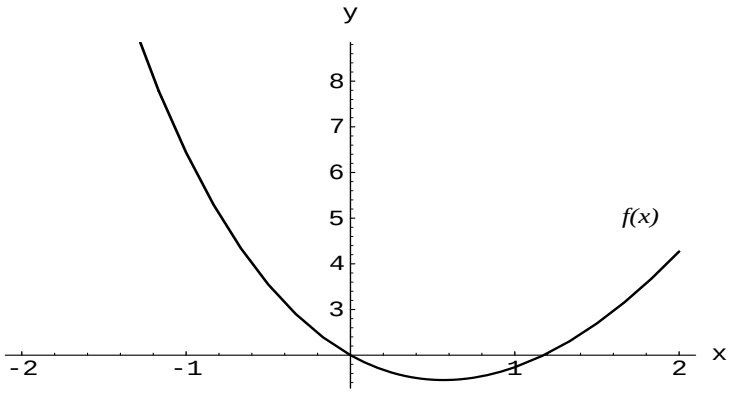


Figura 10.1: Representação gráfica de $y = x^2 + 2e^{-x}$

Um ponto x que verifica as funções de restrição do problema, chama-se *ponto admissível*. Ao conjunto de todos os pontos admissíveis chama-se *região admissível*. Assim, o conjunto

$$C = \{x \in \mathbf{R}^n | c_i(x) = 0, i = 1, \ldots, m \text{ e } c_j(x) \geq 0, j = m+1, \ldots, t\}$$

é a região admissível do problema (10.1).

10.1.2 Características do problema

Uma primeira classificação divide os problemas em *problemas unidimensionais*, para os quais $n = 1$, e *problemas multidimensionais* ($n > 1$).

Exemplo 10.1 A representação gráfica de $f(x)$, função objectivo do problema unidimensional

$$\text{minimizar}_{x \in \mathbf{R}} x^2 + 2e^{-x}$$

define uma curva no espaço $\mathbf{R}^2$. Veja-se a figura 10.1

Exemplo 10.2 Para qualquer função multidimensional (de dimensão n), a equação $y = f(x)$ define uma superfície no espaço $\mathbf{R}^{n+1}$. Por exemplo, no caso em que $n = 2$, os pontos $y = f(x_1, x_2)$ representam uma superfície no espaço a três dimensões (y, x_1, x_2). Para o problema

$$\text{minimizar}_{x \in \mathbf{R}^2} sen(x_1 x_2) + x_1 + x_2,$$

a figura 10.2 apresenta a representação gráfica de $f(x_1, x_2)$.

Se tomarmos y_1 como sendo um valor de $f(x_1, x_2)$, podemos considerar a equação $f(x_1, x_2) = y_1$ como definindo *uma curva* em x_1 e x_2, no plano $y = y_1$. Se um conjunto de curvas planas, obtidas para vários valores de y_i, for desenhado num único plano, obtemos uma figura que é equivalente a um gráfico de *curvas de nível* da função. A figura 10.3 mostra as curvas de nível, também conhecidas por *contornos*, da função $f(x_1, x_2)$.

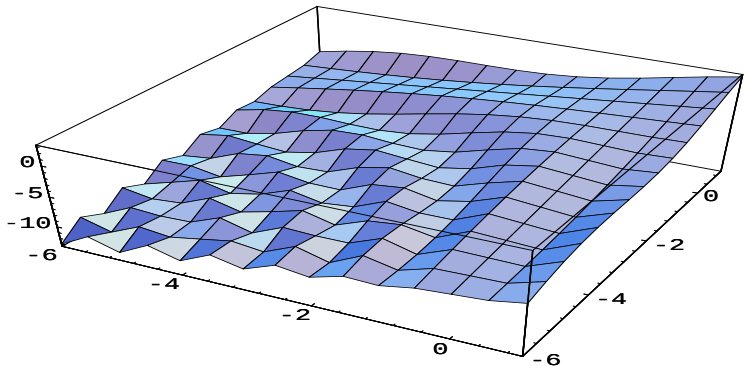


Figura 10.2: Representação gráfica de $y = sen(x_1 x_2) + x_1 + x_2$

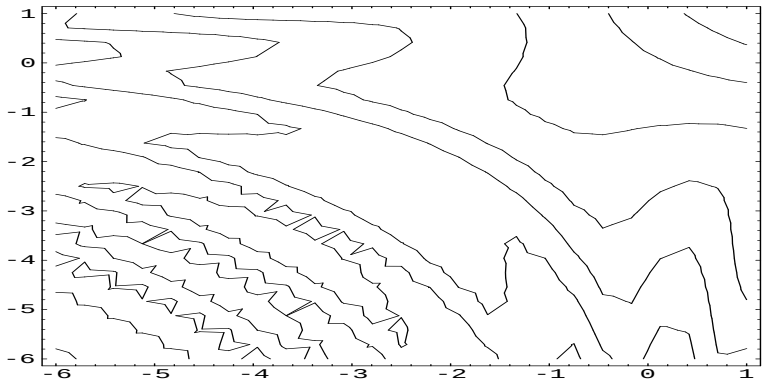


Figura 10.3: Contornos da função $f(x_1, x_2) = sen(x_1 x_2) + x_1 + x_2$

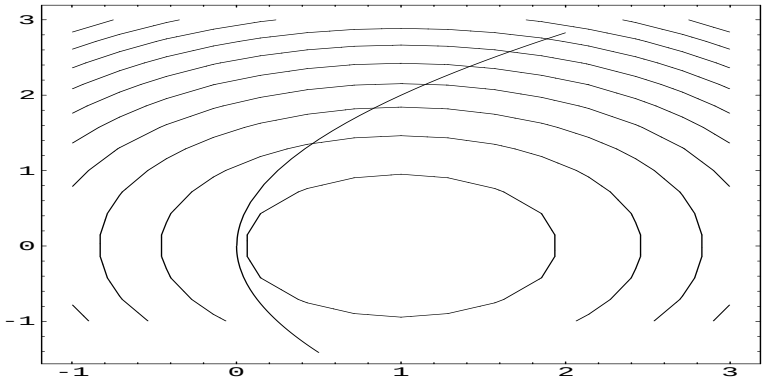


Figura 10.4: Contornos de $f(x_1, x_2) = \frac{1}{2}((x_1 - 1)^2 + x_2^2)$

Em relação à existência ou não de restrições nas variáveis, os problemas de optimização classificam-se em *problemas com restrições*, de que a formulação genérica (10.1) é um exemplo, e *problemas sem restrições nas variáveis*, em que $C = \mathrm{R}^n$, e cuja formulação matemática é

$$\text{minimizar}_{x \in \mathbf{R}^n} f(x). \tag{10.2}$$

Em geral, a solução do problema com restrições nas variáveis não coincide com a solução do problema sem restrições, uma vez que esta pode não pertencer à região admissível. Veja-se o exemplo:

Exemplo 10.3 A solução do problema

$$\text{minimizar}_{x \in \mathbf{R}^2} \frac{1}{2}((x_1 - 1)^2 + x_2^2)$$

é o ponto $x^* = (1, 0)^T$, em que $f^* = 0$. Se adicionarmos ao problema a restrição $c(x) = -x_1 + \frac{1}{4}x_2^2 = 0$, a solução passa a ser o ponto $(0, 0)^T$ com $f^* = \frac{1}{2}$. Veja-se a figura 10.4. O contorno que corresponde a $c(x) = 0$ é bem visível.

Uma classe especial de problemas com restrições é aquela em que os valores aceitáveis para as variáveis são definidos por um conjunto de *restrições lineares*, como, por exemplo, o problema com restrições de igualdade

$$\text{minimizar}_{x \in \mathbf{R}^n} f(x)$$

sujeito a

$$Ax = b,$$

em que a matriz A é do tipo $m \times n$ e $b \in \mathbf{R}^m$. A linha i de A contém os coeficientes da restrição i. Também se podem ter problemas com restrições lineares do tipo desigualdade

$$\text{minimizar}_{x \in \mathbf{R}^n} f(x)$$

sujeito a

$$Ax \geq b.$$

Quando a função $f(x)$ é linear da forma $f(x) = d^T x$, para algum vector constante d de $\mathbf{R}^n$, o problema é designado de *programação linear*[1]. O valor óptimo x^* não é afectado por qualquer termo constante na função objectivo, por isso ele é ignorado na formulação. O vector das primeiras derivadas é o vector constante d e a matriz das segundas derivadas é nula. Para que este problema possa ter uma solução finita é necessário que existam restrições (lineares) nas variáveis, pois f pode tornar-se arbitrariamente grande e negativa.

Um problema de *programação quadrática*[2] é um caso especial de optimização com restrições lineares que surge quando a função f é quadrática,

$$f(x) = \frac{1}{2} x^T G x + d^T x$$

para algum vector constante d e matriz simétrica constante G. O vector das primeiras derivadas é $Gx + d$ e a matriz das segundas derivadas é G.

Alguns problemas têm as restrições com uma forma especial. São limites simples nas variáveis:

$$i_j \leq x_j \leq s_j, \ j = 1, \ldots, n,$$

e qualquer dos limites pode não estar presente.

Um problema que está directamente relacionado com a programação quadrática surge quando se pretende minimizar uma função definida pela $\|.\|_2$ do vector resíduo de um conjunto de equações lineares. O correspondente problema é conhecido por *problema de mínimos quadrados linear*. Pode ter restrições lineares nas variáveis ou não.

Outra classificação importante diz respeito à dimensão do problema. Quando o número de variáveis é 'grande' e a matriz A das restrições é esparsa, o problema diz-se de *grande dimensão*[3]. Existem técnicas especiais para a resolução deste tipo de problemas, uma vez que os métodos mais conhecidos tornam-se muito caros em termos computacionais e em recursos informáticos.

Finalmente, os problemas mais complexos são os *problemas de programação não linear*[4], caracterizados por uma função objectivo $f(x)$ não linear nas variáveis e por funções de

[1] Programação linear foi proposta em 1947 por G.B. Dantzig e tem sido muito usada desde então. No entanto, outros autores escreveram sobre a técnica: Fourier, em 1823, o matemático belga Vallée Poussin, em 1911, e L. Kantorovich, em 1939. Os trabalhos de W. Leontief (1933) e J.V. Newmann (1928 e 1937) também influenciaram o desenvolvimento da programação linear (G.B. Dantzig em Lenstra, Rinnooy Kan e Schrijver (1991)).

[2] O prémio Nobel de 1975 foi entregue aos Prof. L. Kantorovich e T.C. Koopmans pela sua contribuição na teoria da afectação óptima de recursos. Novamente em 1990 o prémio foi para a Programação Matemática. H. Markowitz recebeu-o pela introdução do risco em análises financeiras, que ele formulou como um problema de programação quadrática (M.L. Balinski em Lenstra, Rinnooy Kan e Schrijver (1991)).

[3] As primeiras contribuições no desenvolvimento de métodos para resolver problemas de grandes dimensões começaram a aparecer em 1955, com G. Dantzig, em 1959-1960, com G. Dantzig e P. Wolfe e em 1962, com Benders (G.B. Dantzig em Lenstra, Rinnooy Kan e Schrijver (1991)).

[4] O primeiro trabalho de H.W. Kuhn e A.W. Tucker que usou o termo Programação não linear, foi escrito em 1951. Outros trabalhos que contribuíram para o desenvolvimento da Programação não linear

restrição também não lineares. Se o problema envolver apenas restrições de igualdade, a formulação é

$$\text{minimizar}_{x\in\mathbf{R}^n} f(x) \tag{10.3}$$

sujeito a

$$c_i(x) = 0, i = 1, \ldots, m$$

e, se envolver apenas restrições de desigualdade, tem-se

$$\text{minimizar}_{x\in\mathbf{R}^n} f(x) \tag{10.4}$$

sujeito a

$$c_i(x) \geq 0,\ i = 1, \ldots, p.$$

Em geral, os problemas com restrições de igualdade são mais fáceis de resolver. É possível transformar um problema com restrições de desigualdade num problema com restrições de igualdade. Um dos processos obriga à introdução de mais variáveis no problema. Assim, na formulação (10.4) introduzem-se p novas variáveis y_i, $i = 1, \ldots, p$, de tal forma que as restrições ficam com a seguinte forma

$$c_i(x) - y_i^2 = 0,\ i = 1, \ldots, p.$$

Esta técnica tem o inconveniente de aumentar a dimensão do problema. Passa-se a ter $n + p$ variáveis para determinar. Pode também, nalguns casos, complicar o problema introduzindo novos pontos estacionários, designadamente pontos sela. Se se considera necessária a transformação de restrições do tipo desigualdade em restrições de igualdade, então torna-se mais conveniente a introdução de novas variáveis, conhecidas por *variáveis de folga*, de tal forma que

$$c_i(x) - y_i = 0,\ i = 1, \ldots, p$$

e são adicionadas as restrições

$$y_i \geq 0,\ i = 1, \ldots, p$$

de limites simples, para impôr a não negatividade das novas variáveis.

Optimização com restrições é um problema bem mais difícil de resolver do que o de optimização sem restrições. Por esta razão têm sido desenvolvidas técnicas que reformulam os problemas com restrições nas variáveis e os transformam em problemas sem restrições. Esta transformação exige que se adicionem, à função objectivo, termos que venham em função das restrições, de tal forma que a penalização resultante é tanto maior quanto mais afastado se estiver da região admissível. Por exemplo, tendo como base a formulação matemática em (10.3), a nova função objectivo, de penalização, do problema sem restrições passa a ser

$$P(x, q) = f(x) + \frac{1}{2} q \|c(x)\|_2^2$$

foram publicados por W. Karush em 1939 e F. John em 1948. Foi já nos anos sessenta que G. Zoutendijk, T. Rockafeller, P. Wolfe e R. Cottle desenvolveram a teoria da programação não linear e estenderam as noções de dualidade (G.B. Dantzig em Lenstra, Rinnooy Kan e Schrijver (1991)).

em que q é o parâmetro de penalização, um escalar positivo. A solução, $x^*(q)$, do problema sem restrições ($minimizar P(x,q)$) depende do valor atribuído a q e verifica-se

$$\lim_{q\to\infty} x^*(q) = x^*,$$

sendo x^* a solução do problema original (10.3).

A função de penalização indicada não é a mais eficiente, uma vez que exige a resolução de uma sequência ($q \to \infty$) de problemas de optimização sem restrições para se atingir uma boa aproximação à solução do problema com restrições. Existem outras funções de penalização mais eficientes, para a reformulação de (10.3), e ainda outras, para a reformulação de (10.4). O livro de Fiacco e McCormick (1990) é um excelente elemento de consulta para este tipo de técnicas, conhecidas por *técnicas de penalização sequencial*.

As *funções de penalização exacta* ultrapassam o inconveniente referido no parágrafo anterior, uma vez que a solução $x^*(q)$ do problema sem restrições, coincide com x^*, para valores finitos do parâmetro q. É possível provar que existe um valor de Q positivo, tal que $x^*(q) = x^*$ para $q > Q$. O exemplo mais simples de função de penalização exacta e diferenciável para o problema (10.3) é

$$PE(x,\lambda^*,q) = f(x) - c(x)^T\lambda^* + \frac{1}{2}q\|c(x)\|_2^2$$

sendo λ^* o vector óptimo (de $\mathbf{R}^m$), conhecido por vector dos multiplicadores de Lagrange, que verifica as condições de optimalidade de primeira ordem do problema (10.3) construídas a partir da função Lagrangeana

$$L(x,\lambda) = f(x) - c(x)^T\lambda$$

associada a (10.3).

10.1.3 Classificação de mínimos

A determinação do vector x^* onde a função objectivo f atinge o seu valor mínimo, f^*, é o objectivo principal em optimização. Se f^* é o valor mais baixo que f pode atingir para todos os valores possíveis e admissíveis de x, então x^* é conhecido por *minimizante global*.

Seja x^* um ponto admissível e defina-se o conjunto $V(x^*,\delta)$ como a *vizinhança* δ de x^*, centrada em x^* e de raio $\delta > 0$. Considere-se que $V(x^*,\delta) \subset C$, sendo C a região admissível.

Definição 10.1 : O ponto x^* é um minimizante local forte do problema (10.1) se existir uma quantidade $\delta > 0$ tal que

i) $f(x)$ é definida em $V(x^*,\delta)$,

ii) $f(x^*) < f(x)$ para todo o $x \in V(x^*,\delta)$, com $x \neq x^*$.

Definição 10.2 : O ponto x^* é um minimizante local fraco do problema (10.1), se existir uma quantidade $\delta > 0$ tal que

i) $f(x)$ *é definida em* $V(x^*, \delta)$,

ii) $f(x^*) \leq f(x)$ *para todo o* $x \in V(x^*, \delta)$,

iii) x^* *não é um minimizante local forte*

Na prática, é difícil identificar um minimizante calculado por um método numérico como minimizante global. Apenas se conclui que o ponto encontrado é um minimizante na região procurada. Uma função não linear pode ter vários minimizantes e um deles é o minimizante global. Para a sua identificação torna-se necessário conhecer e calcular todos os minimizantes.

Em alguns problemas não existem minimizantes do tipo descrito. Por exemplo, a função $f(x_1, x_2) = x_1 + x_2^3$ não é limitada inferiormente.

Definição 10.3 : Uma função $f : \mathbb{R} \to \mathbb{R}$ *de uma variável* x *diz-se convexa se para todo o par de valores* x *e* y *de* $\mathbb{R}$ *se verifica*

$$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda) f(y) \tag{10.5}$$

para todo o λ *entre* 0 *e* 1.

Em termos gráficos, uma função f é convexa se a recta que une dois pontos quaisquer do seu gráfico fica acima ou sobrepõe-se ao gráfico de $f(x)$. Se a desigualdade é estrita, f diz-se *estritamente convexa*. A função f é *côncava* se $-f$ é convexa.

Estas definições podem ser aplicadas directamente a problemas a n dimensões, interpretando os argumentos x e y como vectores e a expressão (10.5) em termos de multiplicações de vectores por escalares e adição de vectores. Para funções convexas, os minimizantes locais são também globais.

10.1.4 Condições de optimalidade

O problema mais simples que se considera é o de minimização sem restrições e unidimensional, isto é, em que $n = 1$, equacionado na forma

$$\text{minimizar}_{x \in \mathbb{R}} f(x).$$

Se a função objectivo é continuamente diferenciável até à segunda ordem e um minimizante local de f existe num ponto finito x^*, então as seguintes condições devem ser verificadas.

Condições necessárias para a existência de um minimizante num problema unidimensional sem restrições:

- $f'(x^*) = 0$
- $f''(x^*) \geq 0$.

A primeira condição diz-se de *primeira ordem* e define os *pontos estacionários* da função f. A primeira derivada de f também se anula em pontos que são maximizantes locais e pontos de inflexão. Por exemplo, a função $f(x) = (x-1)^3 - 1$ tem um ponto de inflexão em $x = 1$, com valor $f(1) = -1$. A função f é crescente quer à esquerda quer à direita do ponto.

A segunda condição, porque envolve a segunda derivada de f, é conhecida por *condição de segunda ordem*.

Para garantir que um certo ponto é óptimo, devem verificar-se as condições suficientes.

Condições suficientes para a existência de um minimizante num problema unidimensional sem restrições:

- $f'(x^*) = 0$
- $f''(x^*) > 0$.

Se $f(x)$ é contínua, mas x^* é um ponto de descontinuidade de $f'(x)$, as condições suficientes para que x^* seja um minimizante local forte são:

$$f'(x^* + \varepsilon) > 0 \text{ e } f'(x^* - \varepsilon) < 0$$

em que ε é uma quantidade positiva e próxima de zero.

O problema a n dimensões é equacionado na forma

$$\text{minimizar}_{x \in \mathbf{R}^n} f(x)$$

com $x = (x_1, x_2, \ldots, x_n)^T$.

Passemos à definição das derivadas de primeira e segunda ordens de f.

A derivada parcial de f em ordem a x_i $(i = 1, 2, \ldots, n)$ é

$$\frac{\partial f}{\partial x_i}$$

e é uma função de x. O vector formado pelas n derivadas parciais de f é conhecido por *vector gradiente* de $f(x)$ e será usada a letra g para o identificar,

$$g(x) = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \cdots \\ \frac{\partial f}{\partial x_n} \end{pmatrix}.$$

O vector gradiente é o vector normal ao hiperplano tangente à curva $f(x)$, no ponto x. Este hiperplano pode ser definido pelo vector gradiente e pelo valor de $f(x) \in \mathbf{R}$.

As segundas derivadas parciais de $f(x)$ representam-se por

$$\frac{\partial^2 f}{\partial x_i \partial x_j} \text{ para } i \neq j \text{ e } \frac{\partial^2 f}{\partial x_i^2} \text{ para } i = j$$

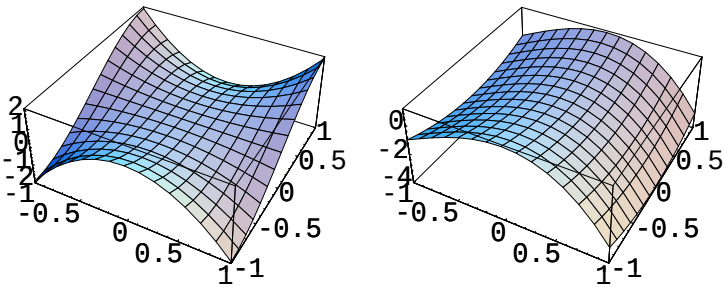


Figura 10.5: Pontos sela: $(0,0)^T$ de $f(x_1,x_2) = x_2(3x_1^2 - x_2^2)$ e $(0,0)^T$ de $f(x_1,x_2) = -x_1^3 + x_2^2 - 3x_1^2$

e são funções de x. Estas n^2 derivadas formam uma matriz quadrada e simétrica de ordem n, conhecida por *matriz Hessiana* e identificada com a letra G,

$$G(x) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_2 \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_n \partial x_1} \\ \frac{\partial^2 f}{\partial x_1 \partial x_2} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_n \partial x_2} \\ & & \cdots & \\ \frac{\partial^2 f}{\partial x_1 \partial x_n} & \frac{\partial^2 f}{\partial x_2 \partial x_n} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix}$$

Condições necessárias para a existência de um minimizante num problema multidimensional sem restrições:

- $\|g(x^*)\|_2 = 0$
- $G(x^*)$ é semidefinida positiva.[5]

A primeira condição é de primeira ordem e define, tal como no caso a uma dimensão, um ponto estacionário. Também, neste caso, um ponto estacionário pode ser um minimizante, um maximizante ou um *ponto sela*[6] (ou *de descanso*). A figura 10.5 mostra dois casos de funções com pontos sela.

A condição de segunda ordem envolve a matriz Hessiana. Se a Hessiana não for semidefinida positiva, o ponto estacionário não é minimizante. Se para um ponto estacionário a Hessiana é definida positiva, das condições suficientes conclui-se que ele é minimizante.

Condições suficientes para a existência de um minimizante num problema multidimensional sem restrições:

[5] Se os determinantes das submatrizes principais de uma matriz são não negativos, a matriz diz-se semidefinida positiva.

[6] As condições necessárias e suficientes para a existência de um ponto sela, de qualquer função diferenciável, com argumentos não negativos, foram formuladas por H.W. Kuhn e A.W. Tucker, em 1951 (H.W. Kuhn em Lenstra, Rinnooy Kan e Schrijver (1991)).

- $\|g(x^*)\|_2 = 0$
- $G(x^*)$ é definida positiva.[7]

As condições necessária e suficiente de segunda ordem para a existência de um máximo no ponto x^* são, respectivamente, $G(x^*)$ semidefinida negativa e definida negativa[8]. Uma matriz que não é definida positiva, semidefinida positiva, definida negativa nem semidefinida negativa, diz-se *indefinida*. Se a Hessiana é indefinida, num ponto estacionário, então esse ponto é *ponto sela*.

Definição 10.4 : Uma função $f \in \mathbb{R}$ diz-se unimodal quando x^* é o único valor da variável para o qual $f(x^*) < f(x)$, para todo o x do intervalo que contém x^*.

Esta definição de mínimo de uma função evita referências às derivadas e aplica-se, portanto, quando a função f tem descontinuidades. A definição pode ser estendida a problemas de dimensão n, substituindo intervalos normais por intervalos a n dimensões. Se uma função é diferenciável e unimodal num intervalo a n dimensões, então a sua Hessiana é definida positiva se e só se ela é estritamente convexa. A equivalência é verdadeira mesmo sem a condição de unimodalidade, embora funções estritamente convexas sejam unimodais.

10.1.5 Índice de algoritmos

A lista dos dez algoritmos desenvolvidos neste Capítulo é a seguinte:

Algoritmo 10.1 Método de procura directa de Fibonacci para calcular um minimizante local de um problema unidimensional

Algoritmo 10.2 Método de procura e interpolação de Davies, Swann e Campey para calcular um minimizante local de um problema unidimensional

Algoritmo 10.3 Critério de paragem dum processo iterativo

Algoritmo 10.4 Método de procura de Rosenbrock para calcular um minimizante local de um problema multidimensional

Algoritmo 10.5 Método do simplex de Nelder-Mead para calcular um minimizante local de um problema multidimensional

Algoritmo 10.6 Critério de Goldstein-Armijo para estimar o comprimento do deslocamento a efectuar ao longo de uma direcção

Algoritmo 10.7 Método das direcções de descida máxima para calcular um minimizante local de um problema multidimensional

Algoritmo 10.8 Método de segurança de Newton para calcular um minimizante local de um problema multidimensional

[7] Se os determinantes das submatrizes principais de uma matriz são positivos, a matriz diz-se definida positiva.

[8] Se os determinantes das submatrizes principais de uma matriz têm sinais alternadamente negativos e positivos, sendo o primeiro negativo, a matriz diz-se definida negativa. Se alguns destes determinantes forem nulos, a matriz é semidefinida negativa.

Algoritmo 10.9 Método do tipo Quasi-Newton para calcular um minimizante local de um problema multidimensional

Algoritmo 10.10 Método dos gradientes conjugados para calcular um minimizante local de um problema multidimensional

Os dois primeiros algoritmos devem ser usados na optimização de problemas a uma dimensão. Para problemas com mais do que uma variável, os algoritmos 10.4, 10.5, 10.7, 10.8, 10.9 e 10.10 são os indicados. Se as derivadas da função $f(x)$ têm descontinuidades em pontos na vizinhança dos óptimos, um dos dois primeiros algoritmos (10.4 ou 10.5) deve ser implementado. Para problemas onde é possível calcular derivadas em qualquer ponto do domínio, os quatro últimos algoritmos são, em geral, mais eficientes. O algoritmo 10.3 define o critério de paragem que deve ser implementado num processo iterativo de optimização em que sejam usadas as derivadas. O algoritmo 10.6 define um critério de escolha do comprimento do deslocamento a efectuar ao longo de direcções de procura, que garante uma redução significativa do valor da função objectivo, de iteração para iteração. O algoritmo 10.8 sendo, em geral, o mais eficiente dos últimos quatro, exige o cálculo da matriz das segundas derivadas de f, que o torna, por vezes, caro em termos computacionais.

10.2 Optimização unidimensional

10.2.1 Introdução

Comecemos por considerar o seguinte problema,

$$\text{minimizar}_{x \in \mathbf{R}} f(x) \tag{10.6}$$

pressupondo que f é unimodal, embora não necessariamente contínua, num intervalo definido por um limite inferior x_i e um limite superior x_s.

Uma condição necessária e suficiente para que x^* seja um minimizante, sem restrições, de $f(x)$, é a de que $f'(x^*) = 0$. Como x^* é um zero da primeira derivada de f, os problemas (10.6) e $f'(x) = 0$ estão relacionados, embora não sejam equivalentes, uma vez que zeros da derivada podem não definir mínimos de f. Assim, é mais aconselhável a implementação de métodos desenvolvidos especificamente para a minimização de funções, do que dos métodos para o cálculo de zeros de funções.

Para a minimização a uma dimensão existem duas classes de métodos. Os *métodos de aproximação*, normalmente aplicáveis a funções contínuas, aproximam a função f por outra função mais simples, que será posteriormente analisada para se determinar o seu mínimo. As funções de aproximação são, em geral, polinomiais de grau baixo. Os *métodos de procura directa* partem de um intervalo, que contém o minimizante, e têm como objectivo reduzi-lo, calculando e comparando valores de f em diversos pontos desse intervalo. Estes métodos podem ser aplicados a qualquer função, desde que ela seja unimodal.

10.2.2 Método de procura Fibonacci

Se uma função é unimodal num intervalo $[x_i, x_s]$, então existe um minimizante $x^* \in [x_i, x_s]$ de tal forma que, dados dois pontos quaisquer x_a e $x_b \in [x_i, x_s]$, com $x_a < x_b$, tem-se

$$\text{se } x_b < x^* \text{ então } f(x_a) > f(x_b),$$

$$\text{se } x_a > x^* \text{ então } f(x_a) < f(x_b).$$

Se a função f é unimodal em $[x_i, x_s]$, é possível reduzir este intervalo comparando os valores da função $f(x)$ em dois pontos interiores do intervalo. Sejam x_a e x_b esses pontos interiores, com $x_a \leq x_b$. O processo de selecção desses pontos interiores é importante, pois o objectivo é reduzir o mais possível o intervalo que contém o minimizante, de iteração para iteração. Para economizar os cálculos de valores da função, um dos pontos interiores do intervalo é transportado para a iteração seguinte. Nesta situação só é preciso calcular uma vez o valor de f, em cada iteração, com excepção da primeira.

Um dos algoritmos, onde a redução dos intervalos é máxima, baseia-se nos números de Fibonacci[9]. O método correspondente chama-se *método de procura Fibonacci*. Os números Fibonacci satisfazem a seguinte relação:

$$F_k = F_{k-1} + F_{k-2} \text{ para } k = 2, 3, \ldots \text{ com } F_0 = F_1 = 1.$$

Os primeiros valores da sequência são: $1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, \ldots$. Dado N, número de vezes que a função f será calculada, usam-se os números Fibonacci F_N, F_{N-1}, $F_{N-2}, \ldots, F_1$ para definir a posição dos pontos interiores x_a e x_b do intervalo, em cada iteração. Assim, para a iteração k tem-se

$$x_a = x_s - \frac{F_{N-k-1}}{F_{N-k}} A_k$$

$$x_b = x_i + \frac{F_{N-k-1}}{F_{N-k}} A_k$$

em que $A_k = x_s - x_i$ é a amplitude do intervalo dessa iteração.

O intervalo da iteração seguinte será $[x_i, x_b]$ ou $[x_a, x_s]$, dependendo dos valores da função $f(x)$ nos dois pontos interiores, x_a e x_b. Assim,

i) se $f(x_b) > f(x_a)$ então o minimizante deve estar entre x_i e x_b e, neste caso, escolhe-se o intervalo $[x_i, x_b]$ para a iteração seguinte. O ponto x_i continua a ser o limite inferior do intervalo e x_b passa a ser o limite superior e será denominado x_s. A amplitude deste novo intervalo é menor do que a amplitude do intervalo da iteração anterior.

ii) se $f(x_b) < f(x_a)$ então o minimizante deve pertencer ao intervalo $[x_a, x_s]$. Este será o intervalo da iteração seguinte. O ponto x_s continua como limite superior e x_a será o limite inferior, passando a ser designado por x_i.

[9] A designação Fibonacci surgiu de Leonardo de Pisa (1175 - 1230) que também era conhecido por Fibonacci (Hämmerlin e Hoffmann (1991)).

Se $f(x_a) = f(x_b)$, pode-se ter já $x_a = x_b$ e o processo chegou ao fim, faltando verificar em que intervalo se encontra o minimizante, no do lado esquerdo ou no do lado direito. Adicionando a x_a uma quantidade positiva e próxima de zero, por exemplo ϵ, comparam-se os valores de $f(x_a)$ e $f(x_a + \epsilon)$.

i) se $f(x_a) > f(x_a + \epsilon)$, o intervalo final que contém o minimizante é o do lado direito;

ii) se $f(x_a) < f(x_a + \epsilon)$, o intervalo final deverá ser o do lado esquerdo.

O ponto médio deste intervalo final poderá ser considerado como a melhor aproximação ao minimizante calculada. O erro desta aproximação é inferior à amplitude deste intervalo final, quer tenha sido seleccionado o do lado esquerdo quer o do lado direito.

Por sua vez, se $f(x_a) = f(x_b)$, mas $x_a \neq x_b$, então o minimizante está entre x_a e x_b. Nesta situação é válido considerar qualquer dos intervalos $[x_i, x_b]$ ou $[x_a, x_s]$, para a iteração seguinte.

Se o intervalo a considerar para a iteração seguinte é o do lado direito $[x_a, x_s]$ e como $x_i \leftarrow x_a$, um dos pontos interiores continua a ser o x_b, que será designado por x_a. O outro ponto interior, x_b, será calculado a partir de

$$x_b = x_i + \frac{F_{N-k-1}}{F_{N-k}} A_k,$$

sendo A_k a amplitude do intervalo agora considerado, $x_s - x_i$, e k o índice da iteração que agora começa.

Se o intervalo para a iteração seguinte é o do lado esquerdo $[x_i, x_b]$ e como $x_s \leftarrow x_b$, o outro ponto interior, x_a, conserva-se mas será designado por x_b. x_a, será, então, calculado a partir de

$$x_a = x_s - \frac{F_{N-k-1}}{F_{N-k}} A_k,$$

com A_k a amplitude do intervalo agora considerado, $x_s - x_i$.

O algoritmo 10.1 apresenta os passos deste processo de procura baseado nos números Fibonacci. O algoritmo Fibonacci é o que origina uma maior redução do intervalo que contém o minimizante, de iteração para iteração. A redução verificada na amplitude do intervalo que contém o minimizante é

$$\frac{A_{N-1}}{A_0} = \frac{1}{F_N},$$

em que A_{N-1} é a amplitude do intervalo depois de N cálculos de f e A_0 a amplitude do intervalo inicial. No entanto, quando comparado com métodos que usam informação relativa às derivadas da função, é menos eficiente.

Algoritmo 10.1 :

1. `ler` N, x_i, x_s `e` ϵ `e fazer` $F_0 = F_1 = 1$
2. `introduzir a função` $f(x)$

3. para $j = 1, \ldots, N-1$ calcular $F_{j+1} = F_{j-1} + F_j$
4. calcular $A = x_s - x_i$, $r = \frac{F_{N-1}}{F_N}$, $x_a = x_s - r\,A$, $x_b = x_i + r\,A$, $f_i = f(x_i)$, $f_a = f(x_a)$, $f_b = f(x_b)$ e $f_s = f(x_s)$ e fazer $k = 1$
5. se $f_a > f_b$ então fazer $x_i = x_a$, $x_a = x_b$, $f_i = f_a$, $f_a = f_b$ e calcular $r = \frac{F_{N-k-1}}{F_{N-k}}$, $x_b = x_i + r(x_s - x_i)$ e $f_b = f(x_b)$
6. senão fazer $x_s = x_b$, $x_b = x_a$, $f_s = f_b$, $f_b = f_a$ e calcular $r = \frac{F_{N-k-1}}{F_{N-k}}$, $x_a = x_s - r(x_s - x_i)$ e $f_a = f(x_a)$
7. se $x_a > x_b$ então fazer $res = x_a$, $x_a = x_b$, $x_b = res$, $res = f_a$, $f_a = f_b$ e $f_b = res$
8. fazer $k = k + 1$
9. se $k < N - 1$ então ir para o passo 5
10. calcular $x_b = x_a + \epsilon$ e $f_b = f(x_b)$
11. se $f_b > f_a$ então fazer $x_s = x_a$ senão fazer $x_i = x_a$
12. calcular $min = \frac{1}{2}(x_i + x_s)$ e $fmin = f(min)$
13. terminar com $x^* \leftarrow min$ e $f(x^*) \leftarrow fmin$.

10.2.3 Implementação. Rotina FIBONA

A rotina FIBONA implementa o algoritmo 10.1. Ao parâmetro ϵ foi dado o valor 10^{-3}.

Exemplo 10.4 Para resolver o problema

$$min_{x \in \mathbf{R}}\, x^2 + 2e^{-x}$$

com $N = 6$ e tomando para intervalo inicial $[0.0, 1.5]$, obtêm-se as seguintes iterações:

k	x_i	x_s	x_a	x_b
0	0.000000	1.500000	0.576923	0.923077
1	0.000000	0.923077	0.346154	0.576923
2	0.346154	0.923077	0.576923	0.692308
3	0.346154	0.692308	0.461538	0.576923
4	0.461538	0.692308	0.576923	0.576923

e como o intervalo final que contém o minimizante é, para $k = 5$ o do lado esquerdo $[x_i, x_a] = [0.461538, 0.576923]$, a aproximação calculada (ponto médio deste intervalo) é 0.519231, com $f(x^*) \approx 1.459557$. O resultado tem uma precisão que não excede a amplitude do último intervalo, que é 0.12 (A_0/F_N).

Se seleccionarmos para N o valor 12, a aproximação conseguida é 0.569742 com uma precisão de 6.5×10^{-3} (A_0/F_N). O valor de f atingido é 1.455949.

Se aumentarmos ainda mais o número de cálculos da função, para $N = 20$, obtém-se $x^* \approx 0.567125$, com uma precisão de 1.4×10^{-4} $(1.5/10946)$ e o valor de f conseguido é de 1.455938.

10.2.4 Métodos de aproximação. Interpolações quadrática e cúbica

Este tipo de métodos baseia-se na aproximação da função $f(x)$, que se pretende minimizar, numa certa região, por outra com uma forma mais simples. A função de aproximação é, em geral, uma forma polinomial de grau baixo, dois ou três. Como o minimizante deste polinómio é fácil de calcular, rapidamente se consegue uma aproximação ao minimizante de $f(x)$. Se esta aproximação não se encontrar suficientemente perto do minimizante, o processo repete-se, considerando-se outra região, provavelmente mais pequena do que a anterior.

Estes métodos de aproximação só podem ser aplicados quando a função $f(x)$ for contínua numa vizinhança do minimizante.

Comecemos pela aproximação quadrática. Para aproximar a função $f(x)$ por um polinómio de grau dois, escolhem-se três valores de x daquela região, x_1, x_2 e x_3 e calculam-se os correspondentes valores de f: $f(x_1)$, $f(x_2)$ e $f(x_3)$. A forma da quadrática é

$$p_2(x) = c_0 + c_1 x + c_2 x^2$$

e o seu minimizante satisfaz

$$\frac{dp_2(x)}{dx} = 0,$$

donde se tira que

$$x^*(q) = -\frac{c_1}{2c_2}. \tag{10.7}$$

Os coeficientes do polinómio c_0, c_1 e c_2 podem ser determinados das equações

$$\begin{cases} c_0 + c_1 x_1 + c_2 x_1^2 &= f(x_1) \\ c_0 + c_1 x_2 + c_2 x_2^2 &= f(x_2) \\ c_0 + c_1 x_3 + c_2 x_3^2 &= f(x_3). \end{cases}$$

Se os pontos x_1, x_2 e x_3 estiverem igualmente distanciados de uma quantidade igual a Δ, e se $x_1 < x_2 < x_3$, da resolução deste sistema e da equação (10.7) tem-se

$$x^*(q) = x_2 + \Delta \frac{f(x_1) - f(x_3)}{2(f(x_3) - 2f(x_2) + f(x_1))}. \tag{10.8}$$

Quando o espaçamento entre os pontos não é constante, da equação (10.7) resulta

$$x^*(q) = \frac{1}{2} \frac{(x_2^2 - x_3^2)f(x_1) + (x_3^2 - x_1^2)f(x_2) + (x_1^2 - x_2^2)f(x_3)}{(x_2 - x_3)f(x_1) + (x_3 - x_1)f(x_2) + (x_1 - x_2)f(x_3)}. \tag{10.9}$$

Se a função de aproximação for um polinómio de grau três,

$$p_3(x) = c_0 + c_1 x + c_2 x^2 + c_3 x^3$$

os pontos estacionários de $p_3(x)$ calculam-se a partir de

$$\frac{dp_3(x)}{dx} = c_1 + 2c_2 x + 3c_3 x^2 = 0,$$

donde

$$x^*(c) = \frac{-c_2 \pm \sqrt{c_2^2 - 3c_1c_3}}{3c_3}. \tag{10.10}$$

O minimizante da cúbica, $x^*(c)$, que será usado para aproximar o minimizante da função $f(x)$ torna a segunda derivada $\frac{d^2p_3(x)}{dx^2} = 6c_3x + 2c_2$ positiva.

Se o polinómio de grau três for construído tendo como base quatro pontos x_1, x_2, x_3 e x_4, e os correspondentes valores de f, $f(x_1), f(x_2), f(x_3)$ e $f(x_4)$, os coeficientes do polinómio calculam-se a partir da resolução do sistema linear

$$\left\{\begin{array}{rcl} c_0 + c_1x_1 + c_2x_1^2 + c_3x_1^3 &=& f(x_1) \\ c_0 + c_1x_2 + c_2x_2^2 + c_3x_2^3 &=& f(x_2) \\ c_0 + c_1x_3 + c_2x_3^2 + c_3x_3^3 &=& f(x_3) \\ c_0 + c_1x_4 + c_2x_4^2 + c_3x_4^3 &=& f(x_4). \end{array}\right. \tag{10.11}$$

Quando o polinómio for construído a partir de dois pontos x_1 e x_2, dos valores da função, $f(x_1)$ e $f(x_2)$, e dos valores da primeira derivada $f'(x_1)$ e $f'(x_2)$, tem-se

$$\left\{\begin{array}{rcl} c_0 + c_1x_1 + c_2x_1^2 + c_3x_1^3 &=& f(x_1) \\ c_0 + c_1x_2 + c_2x_2^2 + c_3x_2^3 &=& f(x_2) \\ c_1 + 2c_2x_1 + 3c_3x_1^2 &=& f'(x_1) \\ c_1 + 2c_2x_2 + 3c_3x_2^2 &=& f'(x_2). \end{array}\right.$$

Substituindo a solução deste sistema na equação (10.10) obtém-se

$$x^*(c) = x_1 + (x_2 - x_1)\{1 - \frac{f'(x_2) + \nu - \eta}{f'(x_2) - f'(x_1) + 2\nu}\} \tag{10.12}$$

em que

$$\eta = 3\frac{f(x_1) - f(x_2)}{x_2 - x_1} + f'(x_1) + f'(x_2)$$

e

$$\nu = (\eta^2 - f'(x_1)f'(x_2))^{\frac{1}{2}}.$$

Como o minimizante, da quadrática ou da cúbica, não coincide com o minimizante da função $f(x)$, o processo terá de ser repetido, originando um processo iterativo. No caso da aproximação quadrática, três pontos são suficientes, podendo ser um deles o minimizante da quadrática, da iteração anterior. Quando se usam os três pontos igualmente distanciados, a distância Δ, entre eles, deve ser menor do que a considerada na iteração anterior. Dos processos conhecidos, o mais eficiente para definir três pontos com espaçamento constante é o referido em 10.2.5, denominado procura de Davies, Swann e Campey.

Se o processo iterativo envolve a aproximação cúbica baseada em quatro pontos, o minimizante encontrado a partir de (10.10) é transportado para a iteração seguinte e a partir dele definem-se os restantes usando também a procura de Davies, Swann e Campey.

Se a aproximação cúbica é baseada em dois pontos e a informação da primeira derivada entra nos cálculos, o minimizante encontrado a partir de (10.12) é um dos pontos que se

transporta para a iteração seguinte. O outro, é um dos que foi usado na iteração anterior. Os dois escolhidos devem verificar o seguinte:

$$\text{se } x_1 < x_2 \text{ então } f'(x_1) < 0 \text{ e } f'(x_2) > 0.$$

Existem algumas objecções à utilização dos métodos de aproximação. Quando a função $f(x)$ é descontínua, o minimizante da aproximação quadrática ou cúbica pode estar muito longe do x^*, uma vez que a aproximação a uma função descontínua pode ser fraca. A implementação de um método de aproximação, por si só, não fornece estimativas da precisão do valor calculado em relação ao minimizante exacto de $f(x)$.

Por sua vez, se o minimizante não pertence ao intervalo definido pelos pontos usados na aproximação, a precisão do minimizante encontrado é, muito provavelmente, fraca, uma vez que não é previsível o comportamento da função f fora do intervalo. Esta dificuldade é facilmente ultrapassada gerando intervalos que contenham o minimizante da função. Uma técnica de procura desses intervalos, que é fácil e de implementação rápida, é introduzida em 10.2.5.

10.2.5 Método de procura de Davies, Swann e Campey

O método de Davies, Swann e Campey é um método de procura que constrói, em cada iteração, três ou quatro pontos igualmente distanciados que definem um intervalo que contém o minimizante, comparando apenas valores de $f(x)$ em diversos pontos.

Esse intervalo, onde se localizam os três ou quatro pontos, é usado posteriormente para aproximar a função, por uma quadrática ou cúbica, com o objectivo de calcular o seu minimizante.

A procura começa com um valor x_1 e uma perturbação δ. A partir do x_1 e no sentido positivo, calcula-se uma sequência de pontos, $x_2, x_3, x_4, x_5, \ldots$ distanciados uns dos outros de, respectivamente δ, 2δ, 4δ, 8δ, ... unidades, até ser encontrado um ponto, para o qual o valor da função tenha aumentado quando comparado com o valor da função no ponto anterior da sequência. Isto é, os pontos de procura definidos são

$$\begin{aligned} x_1 & \\ x_2 &= x_1 + p\,\delta \\ x_3 &= x_2 + p\,2\delta \\ x_4 &= x_3 + p\,4\delta \\ &\ldots \\ x_n &= x_{n-1} + p\,2^{n-2}\delta\,. \end{aligned}$$

com $p = 1$ quando a procura corre no sentido positivo.

A p dá-se o valor -1 se a procura for feita no sentido negativo.

A procura pára no ponto x_k se $f(x_k) > f(x_{k-1})$. Nesta altura, tem-se

$$x_{k-2} < x_{k-1} < x_k, \text{ quando } p = 1$$

ou

$$x_k < x_{k-1} < x_{k-2} \text{ quando } p = -1$$

em que

$$f(x_{k-2}) \geq f(x_{k-1}) \text{ e } f(x_{k-1}) < f(x_k).$$

A distância entre x_k e x_{k-1} é duas vezes a distância entre x_{k-1} e x_{k-2}.

O último valor a calcular deve ser o ponto médio do último intervalo,

$$x_m = \frac{x_k + x_{k-1}}{2}.$$

Tem-se, nesta altura, quatro pontos igualmente espaçados

$$x_{k-2} < x_{k-1} < x_m < x_k, \text{ se } p = 1 \tag{10.13}$$

ou

$$x_k < x_m < x_{k-1} < x_{k-2}, \text{ se } p = -1. \tag{10.14}$$

Quando, a partir do x_1, o valor de $f(x)$ no ponto $x_2 = x_1 + \delta$ for superior a $f(x_1)$, a procura deve voltar-se para o sentido negativo, a começar novamente por x_1. O próximo ponto, na procura, é $\bar{x}_2 = x_1 - \delta$. Se $f(\bar{x}_2)$ for maior do que $f(x_1)$, isso significa que o intervalo definido por $[\bar{x}_2, x_2]$, com x_1 como ponto médio, contém o minimizante desejado. É possível, assim, aproximar a função por um polinómio de grau dois e calcular o minimizante da quadrática, $x^*(q)$, equação (10.8), como se fez em 10.2.4.

Quando a procura decorre quer no sentido positivo quer no negativo, e após serem conhecidos os quatro pontos de (10.13) ou (10.14), a etapa seguinte consiste em aproximar a função, por uma quadrática ou por uma cúbica, no intervalo definido pelos pontos indicados em (10.13) ou (10.14). Para a aproximação cúbica, estes são os quatro pontos que devem ser usados na definição do sistema (10.11), a partir do qual se determina o $x^*(c)$ da equação (10.10).

Para a aproximação quadrática, torna-se necessário seleccionar três dos quatro pontos. Esta selecção é baseada na comparação entre os valores de $f(x)$, nos dois pontos interiores do intervalo. Se

$$f(x_{k-1}) \leq f(x_m) \text{ os três pontos escolhidos são } x_{k-2}, x_{k-1} \text{ e } x_m,$$

$$\text{senão, os pontos seleccionados devem ser } x_{k-1}, x_m \text{ e } x_k.$$

O minimizante da quadrática que passa pelos três pontos, $x^*(q)$, calcula-se usando a equação (10.8).

Se o minimizante encontrado estiver ainda longe do minimizante da função, o processo deve ser repetido. A nova iteração inicia-se pela fase de procura de um intervalo que contenha o minimizante e termina com a aproximação. O recomeço tem como ponto de partida o minimizante $x^*(q)$ ou o $x^*(c)$, da iteração anterior, e deve usar uma perturbação, δ, menor do que a usada na iteração anterior. A constante de redução de δ é $M < 1$. O algoritmo, que corresponde à implementação da procura de Davies, Swann e Campey

seguida de uma aproximação, encontra-se no algoritmo 10.2. O algoritmo termina quando o intervalo entre os pontos, para a aproximação, for inferior a uma precisão pré-definida ε.

Algoritmo 10.2 :

1. introduzir a função $f(x)$
2. ler x_1, ε, δ, M e int ("QUA" ou "CUB"), calcular $f_1 = f(x_1)$ e tol e fazer $k = 0$
3. fazer $k = k + 1$ e $i = 1$
4. calcular $x_2 = x_1 + 2^{i-1}\delta$ e $f_2 = f(x_2)$
5. se $f_2 \leq f_1$ então
   - 5.1. fazer $i = i + 1$
   - 5.2. calcular $x_4 = x_2 + 2^{i-1}\delta$ e $f_4 = f(x_4)$
     - 5.2.1. se $f_4 \leq f_2$ então fazer $x_1 = x_2$, $x_2 = x_4$, $f_1 = f_2$, $f_2 = f_4$ e ir para o passo 5.1
     - 5.2.2. senão calcular $x_3 = x_4 - 2^{i-2}\delta$ e $f_3 = f(x_3)$ e ir para o passo 7
6. senão
   - 6.1. fazer $x_3 = x_2$, $x_2 = x_1$, $f_3 = f_2$, $f_2 = f_1$ e calcular $x_1 = x_2 - 2^{i-1}\delta$ e $f_1 = f(x_1)$
   - 6.2. se $f_1 > f_2$ então
     - 6.2.1. calcular $min = x_2 + 2^{i-1}\delta\frac{f_1 - f_3}{2(f_3 - 2f_2 + f_1)}$ e $fmin = f(min)$
     - 6.2.2. se $2^{i-1}\delta > \varepsilon$ então fazer $x_1 = min$, $f_1 = fmin$, $\delta = M\delta$ e ir para o passo 3, senão ir para o passo 9
   - 6.3. senão
     - 6.3.1. fazer $x_4 = x_2$, $x_3 = x_1$, $f_4 = f_2$ e $f_3 = f_1$
     - 6.3.2. fazer $i = i + 1$
     - 6.3.3. calcular $x_1 = x_3 - 2^{i-1}\delta$ e $f_1 = f(x_1)$
     - 6.3.4. se $f_1 \leq f_3$ então fazer $x_4 = x_3$, $x_3 = x_1$, $f_4 = f_3$, $f_3 = f_1$ e ir para o passo 6.3.2
     - 6.3.5. senão calcular $x_2 = x_1 + 2^{i-2}\delta$ e $f_2 = f(x_2)$ e ir para o passo 7
7. se $int =$ "CUB"então
   - 7.1. para $j = 1, \ldots, 4$
     - 7.1.1. fazer $b_j = f_j$
     - 7.1.2. para $l = 1, \ldots, 4$ calcular $a_{jl} = x_j^{l-1}$
   - 7.2. resolver o sistema de 4 equações $Ac = b$ (algoritmo 3.3.) e colocar a solução em c_0, c_1, c_2, c_3
   - 7.3. se $|c_3| < tol$ então calcular $min = -\frac{c_1}{2c_2}$ e $fmin = f(min)$
   - 7.4. senão, calcular $rai = c_2^2 - 3c_1c_3$

7.4.1. se $rai \leq 0$ então calcular $min = \frac{-c_2}{3c_3}$ e $fmin = f(min)$

7.4.2. senão calcular $min = \frac{-c_2+\sqrt{rai}}{3c_3}$ e $fmin = f(min)$

7.5. se $2^{i-2}\delta > \varepsilon$ então fazer $x_1 = min$, $f_1 = fmin$, $\delta = M\delta$ e ir para o passo 3

8. senão

8.1. se $f_2 > f_3$ então fazer $x_1 = x_2$, $x_2 = x_3$, $x_3 = x_4$, $f_1 = f_2$, $f_2 = f_3$ e $f_3 = f_4$

8.2. calcular $min = x_2 + 2^{i-2}\delta\frac{f_1-f_3}{2(f_3-2f_2+f_1)}$ e $fmin = f(min)$

8.3. se $2^{i-2}\delta > \varepsilon$ então fazer $x_1 = min$, $f_1 = fmin$, $\delta = M\delta$ e ir para o passo 3

9. terminar com $x^* \leftarrow min$ e $f(x^*) \leftarrow fmin$.

10.2.6 Implementação. Rotina DASWCA

A rotina DASWCA implementa o algoritmo 10.2 de Davies, Swann e Campey.

Exemplo 10.5 Na resolução do problema do exemplo 10.4, a rotina DASWCA baseada em interpolação quadrática calcula a aproximação 0.567181 ao fim de 3 iterações, com uma precisão de $\varepsilon = 0.03$. O valor atingido de f é de 1.455938. O processo iterativo é iniciado com $x_1 = 0.0$. Para os parâmetros de entrada, δ e M, são dados respectivamente os valores 0.1 e 0.5.

Se a precisão final exigida for de 0.0001, o processo leva 11 iterações, conseguindo-se $x^* \approx 0.567143$ e $f^* \approx 1.455938$.

Se na fase de interpolação for utilizada uma forma cúbica, a rotina DASWCA leva 3 iterações a atingir 0.567181, com uma precisão de 0.03 e o valor da função correspondente é 1.455938.

Para uma precisão de 0.0001, o processo converge para 0.567143, com $f = 1.455938$, ao fim de 11 iterações. Comparando esta implementação com a correspondente (com os mesmos valores dos parâmetros de entrada) da interpolação quadrática verifica-se que a partir da terceira iteração os resultados intermédios são quase idênticos.

Exemplo 10.6 Considere o seguinte problema

$$min_{x\in\mathbf{R}}|x^2 - 2x|.$$

Para $x_1 = 1$, $\delta = 0.3$ e $M = \frac{1}{3}$ a rotina DASWCA fornece os seguintes resultados:

k	$x^{(k+1)}$	$f(x^{(k+1)})$
1	1.842697	0.289862
2	2.002632	0.005270
3	2.000904	0.001809
4	2.000379	0.000758
5	2.000164	0.000328

e ao fim de 5 iterações o resultado atingido tem uma precisão de 0.01. Esta implementação é baseada em interpolação quadrática.

Repetindo a implementação, mas agora para uma precisão de $\varepsilon = 0.00001$, atinge-se o valor 2.000000 ao fim de 11 iterações, com um valor de f de 0.000001.

Iniciando o processo com $x_1 = 0.25$, obtém-se o valor 0.000000, com $f = 0.000001$, ao fim de 11 iterações, para $\varepsilon = 0.00001$.

Para o mesmo valor inicial e a mesma precisão a interpolação cúbica fornece o valor 0.000000, com $f = 0.000001$, ao fim de 11 iterações.

10.3 Optimização multidimensional

10.3.1 Introdução

As funções não diferenciáveis são, em geral, mais difíceis de minimizar do que as funções continuamente diferenciáveis. Torna-se vantajoso saber distinguir os problemas que têm descontinuidades aleatórias na função e nas derivadas dos que têm descontinuidades bem conhecidas, sobre as quais se conhecem a posição e a natureza.

Existem métodos específicos, baseados na comparação dos valores da função objectivo, para minimizar problemas com descontinuidades na função objectivo e outros para minimizar problemas com descontinuidades bem estruturadas no gradiente.

Se a função objectivo tem poucas descontinuidades no gradiente e se elas não ocorrem na vizinhança do minimizante, então os métodos normalmente usados para problemas continuamente diferenciáveis são os mais eficientes.

Os métodos para a optimização de funções de várias variáveis podem ser agrupados em duas classes, não completamente distintas. Uns, são os *métodos de procura* que usam apenas valores da função, comparando-os, para progredir em direcção ao minimizante; os outros, são os *métodos do gradiente* que utilizam não só os valores de $f(x)$, mas também informação relativa às derivadas, na forma do vector gradiente e/ou da matriz Hessiana. Se apenas usa o vector gradiente o método é caracterizado como sendo de *primeira ordem*; se também usa a matriz Hessiana das segundas derivadas da função, o método é designado por *método de segunda ordem*.

10.3.2 Critério de paragem

Os métodos utilizados na resolução de um problema de optimização não linear multidimensional, sem restrições, são iterativos. O processo gera uma sequência de aproximações $\{x^{(k)}\}$ ao minimizante, x^*, da função objectivo, $f(x)$, e duas questões devem ser levantadas durante a sua implementação.

i) O processo iterativo converge para um valor que minimiza a função, e é esse valor um minimizante global, ou apenas um dos minimizantes locais?

ii) Se o processo iterativo converge, qual é a razão de convergência?

O valor da função objectivo é muitas vezes o único elemento que pode ser usado para se verificar o progresso da minimização. Quando o valor da função não diminui ao longo de um conjunto de iterações, assume-se que não é possível baixar mais o seu valor e que se atingiu um mínimo. No entanto, é quase impossível afirmar se o mínimo atingido é o mínimo global ou um dos locais.

Nenhuma das técnicas iterativas mais usadas em optimização não linear, garante a convergência para um mínimo global, quando existem vários mínimos. Uma procura exaustiva de todos os mínimos encontrados pode, mesmo assim, não responder a essa dúvida.

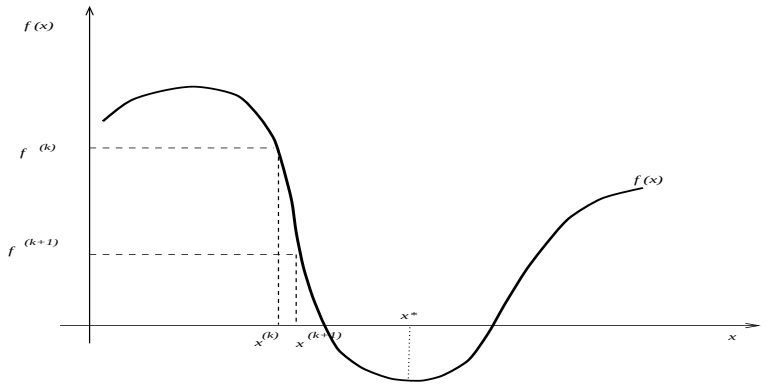


Figura 10.6: Função em **R**

A razão de convergência do método, em termos relativos, é medida normalmente em função do número de iterações que leva até convergir para o mínimo. Os tipos de convergência mais comuns são: *convergência linear*, *convergência superlinear* e a mais rápida, *convergência quadrática*. No Capítulo 2, em 2.1.3. foram já definidos estes três tipos.

Em geral, a sequência $\{x^{(k)}\}$ é infinita, tornando-se necessário utilizar um critério que termine as iterações quando se atinge um valor com uma precisão pré-definida.

Embora $\|x^{(k+1)}-x^{(k)}\|$ seja uma estimativa do erro $\|x^{(k)}-x^*\|$ em $x^{(k)}$, não é aconselhável terminar as iterações quando $\|x^{(k+1)}-x^{(k)}\|$ se torna mais pequeno do que uma certa tolerância, uma vez que a convergência do processo pode não ser superlinear e $\|x^{(k+1)}-x^{(k)}\|$ não é necessariamente uma função monótona decrescente de k. A figura 10.6 mostra um caso, em que a diferença $\|x^{(k+1)}-x^{(k)}\|$ já é pequena, mas a variação $|f^{(k+1)}-f^{(k)}|$ ainda é grande e $x^{(k+1)}$ está ainda longe de x^*.

Torna-se assim necessário usar também outras condições.

Em muitos processos, o valor da função f vai diminuindo, de iteração para iteração. A sequência $\{f^{(k)}\}$ é pois uma sequência monótona decrescente. É razoável supor que $|f^{(k+1)}-f^{(k)}|$ diminui, à medida que k aumenta.

Assim, pode-se usar como condição de convergência o facto de a quantidade $|f^{(k+1)}-f^{(k)}|$ poder tomar valores muito pequenos, a partir de uma certa iteração. A figura 10.7 mostra uma situação em **R** em que a variação $|f^{(k+1)}-f^{(k)}|$ já é pequena, a alteração $\|x^{(k+1)}-x^{(k)}\|$ em x é, no entanto, grande e o minimizante está ainda longe de $x^{(k+1)}$.

Do que foi dito, parece razoável usar, como critério de paragem as duas condições que envolvem as quantidades $\|x^{(k+1)}-x^{(k)}\|$ e $|f^{(k+1)}-f^{(k)}|$.

Se $f(x)$ tem primeiras derivadas parciais contínuas e se esta informação é utilizada no processo, quando se procura um ponto estacionário de $f(x)$, parece razoável verificar se a norma do gradiente, $\|g(x^{(k+1)})\|$, decresce e tende para zero, com k.

Tendo em conta o que foi dito, é possível aconselhar *dois critérios de paragem* para o problema de optimização não linear. O de Himmelblau (proposto em Wolfe (1978))

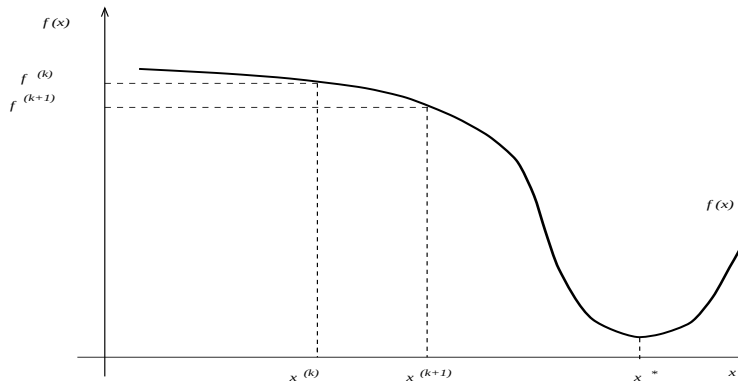


Figura 10.7: Função em **R**

considera as três condições:

$$\frac{\|x^{(k+1)} - x^{(k)}\|}{\|x^{(k+1)}\|} \leq \varepsilon_1 \,, \frac{|f^{(k+1)} - f^{(k)}|}{|f^{(k+1)}|} \leq \varepsilon_2$$

e

$$\|g(x^{(k+1)})\| \leq \varepsilon_3,$$

com $\varepsilon_1, \varepsilon_2$ e ε_3 quantidades pequenas e positivas. A norma 2 de vectores é, neste caso, a mais aconselhável.

O segundo critério (veja-se em Gill e Murray (1976)) é baseado nas condições

$$\|x^{(k+1)} - x^{(k)}\| \leq \varepsilon \left(1 + \|x^{(k+1)}\|\right), \quad |f^{(k+1)} - f^{(k)}| \leq \varepsilon^2 (1 + |f^{(k+1)}|) \tag{10.15}$$

e

$$\|g(x^{(k+1)})\| \leq \varepsilon^{\frac{1}{3}} \left(1 + |f^{(k+1)}|\right), \tag{10.16}$$

em que ε é uma quantidade pequena e positiva que determina a precisão dos resultados obtidos no processo iterativo.

Um critério baseado somente na condição (10.16) não seria adequado, uma vez que existem pontos estacionários que são pontos sela.

O algoritmo 10.3 apresenta o critério de paragem de Gill e Murray, de aplicação mais geral (mesmo para problemas em que o mínimo a atingir é zero), baseado nas três condições apresentadas em (10.15) e (10.16).

Algoritmo 10.3 :

1. `ler` n, k, $f^{(k)}$, $f^{(k+1)}$, ε `e para` $i = 1, \ldots, n$ `ler` $x_i^{(k)}$, $x_i^{(k+1)}$ `e` $g_i^{(k+1)}$
2. `calcular` $nor1 = \sqrt{\sum_{i=1}^{n} (x_i^{(k+1)})^2}$
3. `calcular` $dif = \sqrt{\sum_{i=1}^{n} (x_i^{(k+1)} - x_i^{(k)})^2}$ `e` $diffun = |f^{(k+1)} - f^{(k)}|$

4. se $dif \leq \varepsilon\,(1 + nor1)$ e $diffun \leq \varepsilon^2(1 + |f^{(k+1)}|)$ então

   4.1. calcular $nor2 = \sqrt{\sum_{i=1}^{n}(g_i^{(k+1)})^2}$

   4.2. se $nor2 > (\varepsilon)^{\frac{1}{3}}(1 + |f^{(k+1)}|)$ então ir para o passo 5

   4.3. terminar com convergência=.TRUE.

5. terminar com convergência=.FALSE.

10.3.3 Método de procura de Rosenbrock

Os métodos de procura geram aproximações ao mínimo da função f, calculando e comparando valores da função em diversos pontos da vizinhança da solução. Estas comparações determinam uma *direcção de procura* ao longo da qual se deve procurar o minimizante.

O minimizante pode ser então encontrado através de

i) um *deslocamento de comprimento pré-definido* ao longo dessa direcção;

ii) um *deslocamento de comprimento óptimo*, que minimiza a função ao longo da direcção de procura.

Uma vez que a direcção de procura utilizada não se dirige necessariamente para o minimizante, o processo deve ser repetido, calculando-se uma nova direcção e um deslocamento. O processo deve ser terminado quando for verificado um critério de paragem que garanta que o valor encontrado está muito perto do minimizante.

O método de procura mais simples, baseado num deslocamento pré-definido, considera as direcções de procura paralelas aos eixos coordenados, $e_i, i = 1, \ldots, n$, com e_i a coluna i da matriz identidade de ordem n. O comprimento do deslocamento, h_i, a efectuar ao longo de cada direcção e_i, $i = 1, \ldots, n$, é definido no início de cada iteração. Cada iteração compreende n procuras ao longo das n direcções paralelas aos eixos.

Se, para um novo ponto calculado,

$$x = x^{(k)} + h_i e_i, \;\; i = 1, \ldots, n$$

o valor da função diminui em relação ao valor da função no ponto anterior, esse deslocamento é considerado um *sucesso* e o comprimento desse deslocamento sofre um aumento, para a iteração seguinte. Considera-se que a direcção é boa e pode-se avançar mais depressa. Assim,

$$\text{se } \; f(x) < f(x^{(k)}) \;\; \text{então } \; x^{(k)} \leftarrow x, \; h_i = \alpha h_i, \text{ com } \alpha > 1.$$

Por sua vez, se, no novo ponto calculado, o valor da função aumenta, o deslocamento é considerado um *insucesso*, o ponto é rejeitado e regressa-se ao ponto anterior. Nesta situação, diminui-se o comprimento do deslocamento e altera-se o sentido da procura,

$$\text{se } \; f(x) \geq f(x^{(k)}) \;\; \text{então } h_i = -\beta h_i, \text{ com } 0 < \beta < 1.$$

Os deslocamentos deste tipo apresentam um comportamento ziguezagueante e o processo torna-se muito lento, uma vez que as direcções paralelas aos eixos raramente são paralelas aos eixos principais dos contornos da função.

O *algoritmo de Rosenbrock* é baseado num processo de definição de direcções de procura semelhante ao anterior. Os vectores ortogonais e unitários que são usados, no início do processo iterativo, como direcções de procura, são os eixos coordenados. No entanto, estas direcções de procura são alteradas logo que *se tenha verificado um deslocamento bem sucedido seguido de um insucesso, ao longo de cada uma das direcções.*

Para alterar as direcções de procura, calcula-se, a partir do conjunto antigo de direcções ortonormais, $e_i^{(k)}$, um novo conjunto ortonormal de direcções $e_i^{(k+1)}$, $i = 1, \ldots, n$ para serem usadas a partir da próxima iteração $k+1$.

Do novo conjunto $e_i^{(k+1)}, i = 1, 2, \ldots, n$, a primeira direcção $e_1^{(k+1)}$ é calculada como a resultante dos deslocamentos efectivamente realizados (deslocamentos bem sucedidos), com o conjunto anterior de direcções. As restantes direcções, $e_i^{(k+1)}, i = 2, 3, \ldots, n$, são calculadas usando o *processo de Gram-Schmidt*, que gera vectores ortonormais e que a seguir se descreve.

As repetidas reortonormalizações têm como objectivo alinhar a primeira direcção e_1 com o eixo principal dos contornos de f, a segunda com a melhor direcção perpendicular a e_1, e assim sucessivamente.

Seja y o ponto de partida dos deslocamentos efectuados com o conjunto anterior das n direcções $e_1^{(k)}, e_2^{(k)}, \ldots, e_n^{(k)}$ e $x^{(k)}$ o último ponto aceite.

A direcção resultante dos deslocamentos feitos com as direcções anteriores, é dada por $s_1 = x^{(k)} - y$, ou seja,

$$s_1 = \sum_{i=1}^{n} d_i e_i^{(k)}$$

sendo d_i a soma algébrica dos comprimentos dos deslocamentos bem sucedidos ao longo de $e_i^{(k)}$. Para todos os $i's$, tem-se $d_i \neq 0$. O vector s_1 é normalizado de acordo com,

$$e_1^{(k+1)} = \frac{s_1}{\|s_1\|_2}.$$

Para as restantes direcções $j = 2, \ldots, n$, tem-se

$$s_j = s_{j-1} - d_{j-1} e_{j-1}^{(k)}\,,$$

$$w_j = s_j - \sum_{i=1}^{j-1} (s_j^T e_i^{(k+1)}) e_i^{(k+1)}$$

que depois de normalizadas originam

$$e_j^{(k+1)} = \frac{w_j}{\|w_j\|_2}\,.$$

Se $h_1, h_2, \ldots, h_n$ forem os escalares que definem os comprimentos dos deslocamentos que são efectuados ao longo de cada direcção $e_1, e_2, \ldots, e_n$, o algoritmo de Rosenbrock utiliza o valor $\alpha = 3$ para aumentar o h_i, de uma iteração para a outra, quando o deslocamento é bem sucedido, e $\beta = 0.5$, quando o deslocamento é rejeitado.

Quando se calcula um novo conjunto de direcções de procura, os escalares h_i, $i = 1, \ldots, n$ retomam os valores dados no início do processo iterativo.

Embora o algoritmo original de Rosenbrock não tivesse critério de paragem específico, era apenas baseado no número de cálculos de valores de f, a paragem pode ser feita, seguindo a sugestão indicada em Swann (1972), com base na quantidade

$$R = \frac{\|s_2\|}{\|s_1\|}.$$

Esta quantidade mede o progresso feito ao longo das direcções $e_2, e_3, \ldots, e_n$ relativamente ao progresso total, usando a fracção daquele sobre este. Valores inferiores a 0.3 têm indicado uma certa proximidade do mínimo. $R < 0.03$ tem significado que o mínimo está já perto. Na prática verifica-se, em certas situações, que o processo iterativo não pára embora o mínimo esteja muito perto. É este o critério de paragem usado no algoritmo 10.4 do método de procura de Rosenbrock.

Algoritmo 10.4 :

1. introduzir a função $f(x_1, x_2, \ldots, x_n)$
2. ler n, n_{max}, fazer $k = 0$, $nor = 1$, $sor = 1$ e para $j = 1, \ldots, n$ ler $x_{j\,1}$ e calcular $f_1 = f(x_{1\,1}, x_{2\,1}, \ldots, x_{n\,1})$
3. para $i = 1, \ldots, n$ ler δ_i
4. para $i = 1, \ldots, n$ e para $j = 1, \ldots, n$ se $j = i$ então fazer $e_{j\,i} = 1$ senão fazer $e_{j\,i} = 0$
5. fazer $som = 2n$, $fy = f_1$ e para $i = 1, \ldots, n$ fazer $l_i = 2$, $d_i = 0$ e $h_i = \delta_i$ e $y_i = x_{i\,1}$
6. fazer $m = 1$ e $k = k + 1$
7. calcular $som = som - l_m$, para $j = 1, \ldots, n$ calcular $x_{j\,2} = x_{j\,1} + h_m e_{j\,m}$ e $f_2 = f(x_{1\,2}, x_{2\,2}, \ldots, x_{n\,2})$
8. se $f_2 < f_1$ então
   - 8.1. calcular $d_m = d_m + h_m$, fazer $h_m = 3h_m$, $f_1 = f_2$ e para $j = 1, \ldots, n$ fazer $x_{j\,1} = x_{j\,2}$
   - 8.2. se $l_m = 2$ então fazer $l_m = 1$
9. senão
   - 9.1. fazer $h_m = -\frac{1}{2} h_m$
   - 9.2. se $l_m = 1$ então fazer $l_m = 0$

10. calcular $som = som + l_m$

11. se $som = 0$ então

    11.1. para $j = 1, \ldots, n$ calcular e $s_{j\,1} = \sum_{i=1}^{n} d_i\, e_{j\,i}$

    11.2. para $i = 2, \ldots, n$ e $j = 1, \ldots, n$ calcular $s_{j\,i} = s_{j\,i-1} - d_{i-1} e_{j\,i-1}$

    11.3. calcular $nor = \sqrt{\sum_{j=1}^{n} s_{j1}^2}$, $sor = \sqrt{\sum_{j=1}^{n} s_{j2}^2}$ e para $j = 1, \ldots, n$ calcular $e_{j\,1} = \frac{s_{j\,1}}{nor}$

    11.4. para $i = 2, \ldots, n$

        11.4.1. para $r = 1, \ldots, i-1$ calcular $\alpha_r = \sum_{j=1}^{n} s_{j\,i}\, e_{j\,r}$

        11.4.2. para $j = 1, \ldots, n$ calcular $w_j = s_{j\,i} - \sum_{r=1}^{i-1} \alpha_r e_{j\,r}$

        11.4.3. calcular $wor = \sqrt{\sum_{j=1}^{n} w_j^2}$ e para $j = 1, \ldots, n$ calcular $e_{j\,i} = \frac{w_j}{wor}$

    11.5. ir para o passo 5

12. senão

    12.1. se $m = n$ então

        12.1.1. se $\frac{sor}{nor} \geq 0.03$ então

            12.1.1.1. se $k \geq n_{max}$ então parar, o processo não está a convergir

            12.1.1.2. ir para o passo 6

    12.2. senão fazer $m = m + 1$ e ir para o passo 7

13. terminar com $x^* \leftarrow (x_{j\,1}, j = 1, \ldots, n)$ e $f(x^*) \leftarrow f_1$.

10.3.4 Implementação. Rotina ROSENB

A implementação do algoritmo 10.4 origina a rotina ROSENB.

Exemplo 10.7 Considere o seguinte problema

$$min_{x \in \mathbf{R}^2} \text{máx}[|x_1|, |x_2|].$$

Iniciando o processo iterativo com o vector inicial $(x_{11}, x_{21}) = (-2, 0)$ e tomando $(\delta_1, \delta_2) = (1, 1)$, a rotina ROSENB calcula o vector $(-1, 0)^T$, com $f = 1$, na $1^{a.}$ iteração, durante a procura ao longo da $1^{a.}$ direcção; rejeita todos os outros deslocamentos seguintes, volta a aceitar na $4^{a.}$ iteração, ao longo da $1^{a.}$ direcção, o ponto $(-0.25, 0)^T$, com $f = 0.25$; volta a rejeitar os deslocamentos que se seguem, até chegar à iteração 9 em que calcula o ponto $(-0.109375, 0)^T$, com $f = 0.109375$. Rejeita novamente uma série de tentativas e volta a aceitar, na iteração 12, o ponto $(-0.003906, 0)^T$ e depois nas iterações 19, com o ponto $(0.001038, 0)^T$, 23 com o ponto $(-0.000816, 0)^T$, 27 com o ponto $(-0.000121, 0)^T$, 32 com o ponto $(0.000009, 0)^T$ e $f = 0.000009$. O processo está a convergir mas é muito lento e nunca tem possibilidade de calcular novas direcções de procura, uma vez que nenhum deslocamento feito ao longo da segunda direcção é aceite.

Exemplo 10.8 No cálculo do $min\, f(x_1, x_2)$ para

$$f(x_1, x_2) = (x_1 + 1)^2 + (x_2 + 1)^2,$$

com $(x_{11}, x_{21}) = (1, 0)$ e $\delta = (1, 1)^T$ a implementação origina o seguinte comportamento:

	na iteração	nas direcções	novas direcções de procura
aceita	2	1 e 2	
aceita	3	1	
-	no meio da 4	calcula:	$(-.970143, -.242536)^T$; $(.242536, -.970143)^T$
aceita	7	2	
aceita	9	1	
-	no meio da 10	calcula:	$(.000000, -1)^T$; $(1, .000000)^T$
aceita	13	1	
...			

Na iteração 13, o ponto calculado é $(-1.000000, -1.007694)^T$ com $f = 0.0000591992$. Na iteração 21 aceita o ponto $(-1, -1.001835)^T$ com $f = 0.000003$.

Se, na implementação do processo de Gram-Schmidt, for utilizada a norma infinita, em vez da norma 2, o algoritmo de Rosenbrock termina ao fim de 35 iterações, com o ponto $(-1.000098, -0.999156)^T$ e $f = 0.000001$. Na condição de paragem foi usado o valor 0.03. Se em vez deste, for usado 0.3, o processo termina mais cedo, ao fim de 20 iterações, com uma aproximação não tão boa $(-1.000092, -0.992318)^T$ e $f = 0.000059$.

Exemplo 10.9 No cálculo de

$$\min f(x_1, x_2) = x_1^2 + x_2^2 - \frac{1}{3}x_1 x_2,$$

com $(x_{11}, x_{21}) = (1, 1)$ e $\delta = (1, 1)^T$ a implementação calcula novas direcções de procura, no fim da iteração 3. Além de aceitar o ponto $(0.5, 1)^T$, na segunda iteração, ao longo da $1^{a\cdot}$ direcção, e o ponto $(0.5, 0.5)^T$ ao longo da $2^{a\cdot}$ direcção, só volta a aceitar pontos ao longo da $1^{a\cdot}$ direcção e nas iterações 4, 8, 12, 16, 21, 25, 30 e 34. Aqui, o ponto calculado é $(-0.000027, -0.000027)^T$ com $f = 0.000000$.

Se o processo iterativo for iniciado com $(0.5, 0.5)^T$, o ponto aceite na $1^{a\cdot}$ iteração, ao longo da segunda direcção é $(0, 0)^T$ com $f = 0$.

10.3.5 Método de procura de Nelder - Mead

As vantagens mais significativas deste método residem na sua fácil programação e no processo de definição, através de expressões algébricas muito simples, de vários pontos auxiliares que são *aceites* ou *rejeitados* de acordo com os correspondentes valores da função $f(x)$.

Para um problema de $\mathbf{R}^n$,

$$min_{x \in \mathbf{R}^n} f(x)\,,$$

cada iteração do *algoritmo do simplex de Nelder -Mead*, define $n+1$ pontos, x_1, x_2, $x_3, \ldots,$ x_{n+1}, a partir dos quais vai calculando outros *pontos auxiliares*, ao longo do processo.

Os $n+1$ pontos usados no início de cada iteração, $x_1, x_2, \ldots, x_{n+1}$, são considerados os vértices de um simplex a n dimensões. Por exemplo, em $\mathbf{R}^2$, os $n+1(=3)$ pontos determinam um triângulo. Em cada iteração, designe-se o simplex com vértices $x_1, x_2, \ldots, x_{n+1}$ por

$$S =< x_1, x_2, \ldots, x_{n+1} > \ .$$

Se estes pontos forem ordenados de acordo com os correspondentes valores da função, isto é, de tal forma que

$$f(X_1) \leq f(X_2) \leq \cdots \leq f(X_{n+1}),$$

os pontos passam a ser denotados por:

$$X_1, X_2, \ldots, X_{n+1}\,.$$

Assim, X_1 é o melhor vértice do simplex, ..., X_n é o segundo pior e X_{n+1} é o pior vértice. Um *ponto auxiliar* é *aceite* ou *rejeitado* dependendo do correspondente valor da função, quando comparado com os valores de $f(X_1), f(X_n)$ e $f(X_{n+1})$, ou seja, valores da função do melhor (com valor da função mais baixo) e dos dois piores pontos (com valores da função mais altos).

Os *pontos auxiliares* podem definir vértices reflectidos, expandidos, contraídos para o interior, contraídos para o exterior ou de um simplex encolhido.

- Um *vértice reflectido*, obtém-se reflectindo o pior vértice do simplex, X_{n+1}, no sentido do centróide dos restantes n vértices, da seguinte maneira:

  $$x_r = (1+\alpha)\overline{x} - \alpha X_{n+1}$$

  em que $\alpha = 1$ e $\overline{x}$ é o centróide definido por

  $$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} X_i\,.$$

1. Se $f(x_r) < f(X_n)$ então o vértice x_r é considerado melhor do que o penúltimo da ordenação e

   i) se $f(x_r) \geq f(X_1)$ então este vértice reflectido é aceite. A próxima iteração inicia-se a partir do simplex $< X_1, X_2, \ldots, X_n, x_r >$;

   ii) se $f(x_r) < f(X_1)$ então o ponto x_r é mesmo muito bom e passa-se à expansão do simplex:

- Um *vértice expandido* calcula-se a partir de:

  $$x_e = \gamma x_r + (1-\gamma)\overline{x}$$

  com $\gamma = 2$ e

   i) se $f(x_e) < f(X_1)$ então este vértice expandido é aceite. A próxima iteração inicia-se a partir do simplex $< X_1, X_2, \ldots, X_n, x_e >$;

   ii) se $f(x_e) \geq f(X_1)$ então aceita-se o vértice reflectido e a próxima iteração inicia-se a partir do simplex $< X_1, X_2, \ldots, X_n, x_r >$.

2. Por sua vez, se $f(x_r) \geq f(X_n)$ então o ponto reflectido é pior do que o X_n e deve-se calcular um vértice contraído.

   i) Se $f(x_r) \geq f(X_{n+1})$ então o pior dos vértices, X_{n+1}, é pelo menos tão bom quanto o vértice reflectido. A contracção deve ser para o *interior* do simplex:

- Um *vértice contraído para o interior* calcula-se a partir de

$$x_c = \beta X_{n+1} + (1-\beta)\overline{x}$$

com $\beta = 1/2$ e

   a) se $f(x_c) < f(X_n)$, aceita-se este vértice contraído para o interior, e o simplex para a próxima iteração é $< X_1, X_2, \ldots, X_n, x_c >$;
   b) se $f(x_c) \geq f(X_n)$, o passo seguinte consiste em encolher o simplex.

   ii) Se o vértice reflectido é melhor do que o pior vértice do simplex, $f(x_r) < f(X_{n+1})$, calcula-se, então, um vértice *contraído* para o exterior:

- Um *vértice contraído para o exterior* calcula-se a partir de

$$\hat{x}_c = \beta x_r + (1-\beta)\overline{x}$$

com $\beta = 1/2$ e

   a) se $f(\hat{x}_c) < f(X_n)$, aceita-se este vértice. A iteração seguinte deve ser iniciada a partir do simplex $< X_1, X_2, \ldots, X_n, \hat{x}_c >$.
   b) se o vértice contraído para o exterior, $\hat{x}_c$, é pelo menos pior do que o X_n, o passo seguinte consiste em encolher o simplex.

- *Encolher o simplex* consiste em substituir cada um dos vértices X_i, $i = 2, \ldots, n+1$, pelo ponto médio do segmento que une esse X_i a X_1, isto é:

$$x_i \leftarrow \frac{X_i + X_1}{2}.$$

  Os correspondentes valores de f, $f(x_i)$, $i = 2, \ldots, n+1$, são calculados e juntamente com o $f(X_1)$ procede-se à ordenação do simplex $< X_1, x_2, \ldots, x_{n+1} >$ para iniciar nova iteração.

A figura 10.8 apresenta os diversos tipos de vértices localizados em relação a um simplex $< X_1, X_2, X_3 >$ de $\mathbf{R}^2$.

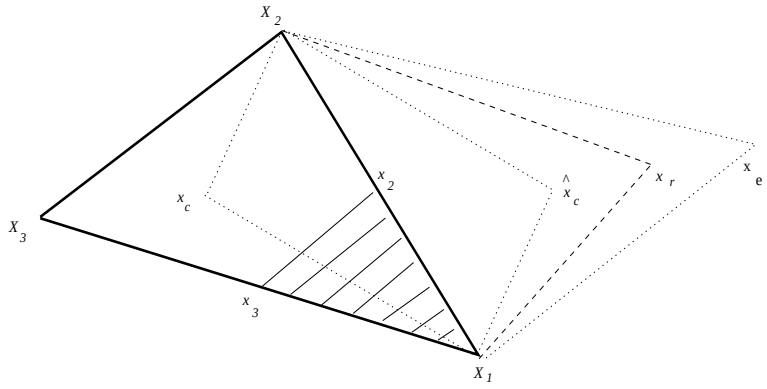


Figura 10.8: Vértices de um simplex em $\mathbf{R}^2$

Nota 1: Depois de encontrado o simplex para a iteração seguinte, procede-se à sua ordenação. Com o simplex já ordenado vai verificar-se o critério de paragem para saber se é necessário fazer mais iterações.

O critério de paragem deste processo iterativo usa uma medida do tamanho relativo do simplex. Se

$$\frac{1}{\Delta}\,\text{máx}_{2\le i\le n+1}\|X_i - X_1\| \le \varepsilon \tag{10.17}$$

com ε uma quantidade pequena e positiva, então o minimizante local está muito perto e o processo iterativo pode ser parado. Caso contrário, o processo iterativo deve continuar.

O valor de Δ é definido por

$$\Delta = \text{máx}\,(1, \|X_1\|)$$

sendo $\|.\|$ a norma 2 de um vector.

Nota 2: Quando o critério de paragem é verificado e o processo termina, usa-se o melhor vértice do simplex X_1 como a melhor aproximação calculada ao minimizante.

O algoritmo 10.5 corresponde ao método do simplex de Nelder-Mead.

Algoritmo 10.5 :

1. `ler` n, α, β, γ, ε, n_{max} `e fazer` $k = 1$
2. `introduzir a função` $f(x_1, x_2, \ldots, x_n)$
3. `para` $i = 1, \ldots, n+1$ `ler` $(x_{j\,i}, j = 1, \ldots, n)$ `e calcular` $f_i = f(x_{1\,i}, x_{2\,i}, \ldots, x_{n\,i})$
4. `ordenar os vértices do simplex:` `para` $i = 1, \ldots, n$ `e para` $l = i+1, \ldots, n+1$ `se` $f_i > f_l$ `então fazer` $gua_1 = f_i$, $f_i = f_l$, $f_l = gua_1$ `e para` $j = 1, \ldots, n$ `fazer` $gua_j = x_{j\,i}$, $x_{j\,i} = x_{j\,l}$ `e` $x_{j\,l} = gua_j$
5. `calcular o centróide:` `para` $j = 1, \ldots, n$ `calcular` $med_j = \frac{1}{n}\sum_{i=1}^{n} x_{j\,i}$
6. `para` $j = 1, \ldots, n$ `calcular` $ref_j = (1+\alpha)med_j - \alpha x_{j\,n+1}$ `e fazer` $xk_j = ref_j$, `calcular` $fref = f(ref_1, ref_2, \ldots, ref_n)$ `e fazer` $fk = fref$

7. se $fref < f_n$ então

   7.1. se $fref < f_1$ então

      7.1.1. para $j = 1, \ldots, n$ calcular $exp_j = \gamma ref_j + (1-\gamma) med_j$ e $fexp = f(exp_1, exp_2, \ldots, exp_n)$

      7.1.2. se $fexp < f_1$ então para $j = 1, \ldots, n$ fazer $xk_j = exp_j$ e $fk = fexp$

8. senão

   8.1. para $j = 1, \ldots, n$ fazer $aux_j = x_{j\,n+1}$ e fazer $faux = f_{n+1}$

   8.2. se $fref < faux$ então para $j = 1, \ldots, n$ fazer $aux_j = ref_j$

   8.3. para $j = 1, \ldots, n$ calcular $con_j = \beta aux_j + (1-\beta) med_j$ e $fcon = f(con_1, \ldots, con_n)$

   8.4. se $fcon < f_n$ então para $j = 1, \ldots, n$ fazer $xk_j = con_j$ e $fk = fcon$

   8.5. senão encolher o simplex:

      8.5.1. para $i = 2, \ldots, n$ e $j = 1, \ldots, n$ calcular $x_{j\,i} = \frac{1}{2}(x_{j\,1} + x_{j\,i})$ e $f_i = f(x_{1\,i}, x_{2\,i}, \ldots, x_{n\,i})$

      8.5.2. para $j = 1, \ldots, n$ calcular $xk_j = \frac{1}{2}(x_{j\,1} + x_{j\,n+1})$ e $fk = f(xk_1, \ldots, xk_n)$

9. para $j = 1, \ldots, n$ fazer $x_{j\,n+1} = xk_j$ e $f_{n+1} = fk$

10. ordenar os vértices do simplex: para $i = 1, \ldots, n$ e para $l = i+1, \ldots, n+1$ se $f_i > f_l$ então fazer $gua_1 = f_i$, $f_i = f_l$, $f_l = gua_1$, e para $j = 1, \ldots, n$ fazer $gua_j = x_{j\,i}$, $x_{j\,i} = x_{j\,l}$ e $x_{j\,l} = gua_j$

11. calcular $nor = \sqrt{\sum_{j=1}^{n} x_{j\,1}^2}$, $\Delta = \max\{1, nor\}$, para $i = 2, \ldots, n+1$ calcular $dis_i = \sqrt{\sum_{j=1}^{n} (x_{j\,i} - x_{j\,1})^2}$ e $max = \max\{dis_2, dis_3, \ldots, dis_{n+1}\}$

12. se $\frac{max}{\Delta} > \varepsilon$ então

   12.1. se $k \geq n_{max}$ então parar, o processo não convergiu

   12.2. fazer $k = k+1$ e ir para o passo 5

13. terminar com $x^* \leftarrow (x_{j\,1}, j = 1, \ldots, n)$ e $f(x^*) \leftarrow f_1$.

10.3.6 Implementação. Rotina NELMEA

A rotina NELMEA corresponde à implementação do algoritmo 10.5 no qual são atribuídos os seguintes valores: $\alpha = 1, \beta = 0.5$ e $\gamma = 2$.

Exemplo 10.10 Na resolução do problema do exemplo 10.8 a rotina NELMEA calcula a aproximação $(-1.000000, -1.000000)^T$, com $f = 0.000000$, ao fim de 16 iterações, quando o conjunto inicial de vectores é

$$x_1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad x_2 = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \text{e} \quad x_3 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

e $\varepsilon = 0.01$. O último conjunto de pontos, já ordenados, é

$$X_1 = \begin{pmatrix} -1.0 \\ -1.0 \end{pmatrix}, \quad X_2 = \begin{pmatrix} -1.004958 \\ -1.012105 \end{pmatrix} \text{ e } X_3 = \begin{pmatrix} -1.021742 \\ -1.003841 \end{pmatrix}.$$

O melhor ponto é X_1, que deve ser considerado como a melhor aproximação ao minimizante.

Quando é dado a ε o valor 10^{-3}, o processo converge ao fim de 23 iterações para o mesmo valor. O último conjunto de pontos é

$$X_1 = \begin{pmatrix} -1.0 \\ -1.0 \end{pmatrix}, \quad X_2 = \begin{pmatrix} -0.999149 \\ -1.000860 \end{pmatrix} \text{ e } X_3 = \begin{pmatrix} -1.000684 \\ -1.001549 \end{pmatrix}.$$

Implementações mais eficientes deste problema podem ser encontradas nos exemplos 10.12, 10.14, 10.16 e 10.18.

Exemplo 10.11 Para calcular

$$min_{x \in \mathbf{R}^2}\, x_1^2 + x_2^2 - \frac{1}{3} x_1 x_2$$

(veja-se o exemplo 10.9) e considerando o conjunto inicial

$$x_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad x_2 = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \text{ e } x_3 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

obtém-se, ao fim de 31 iterações, como melhor aproximação o vector $(-0.000020, -0.000023)^T$, com $f = 0.000000$. A paragem deve-se ao facto da condição em (10.17) ter sido verificada, para $\varepsilon = 10^{-4}$.

Se o valor da tolerância, na condição de paragem, for aumentado, o processo termina mais cedo, embora ainda longe da solução. Assim, para $\varepsilon = 10^{-2}$ o processo demora 17 iterações e a melhor aproximação calculada é $(-0.004022, -0.003112)^T$ com $f = 0.000022$.

10.3.7 Métodos de procura unidimensional

Os métodos do gradiente, para a resolução de problemas não lineares de optimização sem restrições, pertencem à classe dos *métodos iterativos*, e são caracterizados por uma equação iterativa que, a partir de uma aproximação inicial $x^{(1)}$, ao minimizante x^* da função $f(x)$, constrói uma sequência $\{x^{(k)}\}, k = 1, 2, \ldots$ de aproximações a x^*.

Impostas certas condições em $x^{(1)}$ e f, a sequência de aproximações converge para x^*.

Definição 10.5 : Seja $f : \mathbb{R}^n \to \mathbb{R}$ uma função com primeiras derivadas parciais contínuas num ponto $\overline{x} \in \mathbb{R}^n$ e seja $d \in \mathbb{R}^n$ um vector dado não nulo. Seja $g(\overline{x})$ o vector gradiente de $f(x)$, calculado no ponto $\overline{x}$.

O vector d define-se como sendo uma direcção de descida, *em relação a $f(x)$, em $\overline{x}$, se*

$$g(\overline{x})^T d < 0.$$

Se a direcção d for de descida, o valor da função f diminui à medida que nos afastamos de $\overline{x}$, ao longo dessa direcção. Isto é, existe um escalar α positivo, tal que

$$f(\overline{x} + \alpha d) < f(\overline{x}). \tag{10.18}$$

O escalar α dá-nos o *comprimento do deslocamento* que devemos efectuar ao longo da direcção, d.

O procedimento que calcula o valor do escalar α chama-se *procura unidimensional*.

Estas ideias sugerem uma *classe* de métodos para minimizar $f(x)$. Estes métodos, conhecidos por *métodos das direcções de descida* podem ser descritos, na generalidade, pelos seguintes passos:

1. *Dar* $x^{(1)}$ *e calcular* $f(x^{(1)})$ *e* $g(x^{(1)})$. *Fazer* $k = 1$
2. *Calcular o vector direcção* $d^{(k)}$ *de tal forma que* $g(x^{(k)})^T d^{(k)} < 0$
3. *Calcular o escalar* $\alpha^{(k)}$ *de tal modo que* $f(x^{(k)} + \alpha^{(k)} d^{(k)}) < f(x^{(k)})$
4. *Calcular* $x^{(k+1)} = x^{(k)} + \alpha^{(k)} d^{(k)}$, $f(x^{(k+1)})$ *e* $g(x^{(k+1)})$.
5. *Se* $x^{(k+1)}$ *satisfaz o critério de paragem, terminar.*
6. *(senão) Fazer* $k = k + 1$ *e voltar para 2.*

Existe uma grande variedade de métodos de optimização para calcular o vector direcção $d^{(k)}$ e o escalar $\alpha^{(k)}$ que satisfazem as condições dos passos 2. e 3. deste algoritmo.

Por vezes é mais vantajoso calcular um valor de α que minimiza, localmente, a função $f(x)$ ao longo da direcção $d^{(k)}$. Estes cálculos originam um *valor óptimo* do comprimento do deslocamento a efectuar ao longo de $d^{(k)}$ e ao processo é costume chamar *procura unidimensional exacta*.

A condição de minimizar $f(x^{(k)} + \alpha d^{(k)})$, em relação a α, define um problema de minimização unidimensional, e é muito mais restrita do que a condição imposta pelo passo 3. do algoritmo.

Para determinar $\alpha^{(k)}$, na iteração k, que minimiza f como uma função apenas de α

$$\min_{\alpha \in \mathbf{R}} \phi(\alpha) \equiv f(x^{(k)} + \alpha d^{(k)}), \tag{10.19}$$

de acordo com a condição necessária e suficiente de primeira ordem, resolve-se

$$\phi'(\alpha) = 0. \tag{10.20}$$

Embora um valor de $\alpha^{(k)}$ que satisfaz $f(x^{(k)} + \alpha^{(k)} d^{(k)}) = min_\alpha f(x^{(k)} + \alpha d^{(k)})$ produz uma diminuição óptima no valor da função f, em cada iteração do algoritmo com direcções de descida, a procura unidimensional exacta é na maior parte dos casos difícil de implementar.

- Só em casos muito particulares é que a equação $\phi^{'}(\alpha) = 0$ pode ser resolvida exactamente.

  Nos outros casos usam-se algoritmos de aproximação polinomial, nomeadamente quadrática ou cúbica, para determinar uma aproximação ao valor óptimo de α. Estes

métodos para estimar $\alpha^{(k)}$ baseiam-se na aproximação da função $\phi(\alpha)$ por um polinómio em α de grau dois ou três, determinando depois, analiticamente, o minimizante dessa aproximação polinomial. O algoritmo de Davies, Swann e Campey (veja-se o algoritmo 10.2) é dos algoritmos mais conhecidos e usados para estimar o $\alpha^{(k)}$. A fase de procura, de um intervalo que contenha o minimizante, pode obrigar a muitos cálculos de f, tornando o algoritmo pouco eficiente.

A maior parte do trabalho de computação necessário para implementar muitos dos algoritmos com direcções de descida reside na determinação do $\alpha^{(k)}$.

A eficiência computacional da procura unidimensional é determinante para o desempenho do algoritmo de descida. Por esta razão, têm sido dedicados esforços no desenvolvimento de técnicas eficazes e rápidas de procura unidimensional.

Por vezes, os métodos mais simples e menos precisos são preferidos, uma vez que uma redução, considerada significativa, no valor da função é suficiente para garantir convergência para um minimizante.

Um dos critérios mais simples, para calcular um valor de $\alpha^{(k)}$ positivo, que origina uma *redução significativa* no valor de ϕ, é conhecido por *critério de Armijo*. Pressupondo que a direcção de procura é de descida, gera-se uma sequência de valores,

$$\{w^j \alpha\}, \quad j = 0, 1, 2, \ldots$$

a partir de um valor inicial α. Com a constante de redução $w < 1$, os valores vão sucessivamente tornando-se mais pequenos.

O valor inicial mais comum é a unidade.

O primeiro elemento da sequência que origina a *redução em* f

$$f(x^{(k)}) - f(x^{(k)} + w^j \alpha d^{(k)}) \geq -\mu_1 w^j \alpha g(x^{(k)})^T d^{(k)}, \tag{10.21}$$

definida como *significativa*, é aceite na iteração k

$$\alpha^{(k)} \leftarrow w^j \alpha.$$

Esta condição (10.21) é conhecida por *condição de Armijo*.

O factor de redução na sequência $\{w^j \alpha\}$ é em geral $w = 0.5$. Outros valores podem aqui ser usados, e até para o valor de partida não há regra que determine exactamente qual o mais apropriado.

Valores exageradamente grandes não são aceites pela condição de Armijo, embora possam passar valores muito pequenos. Para ultrapassar este inconveniente, é vantajoso adicionar outra condição para aceitação de um valor de $\alpha^{(k)}$. A condição adicional é definida por

$$f(x^{(k)}) - f(x^{(k)} + w^j \alpha d^{(k)}) \leq -\mu_2 w^j \alpha g(x^{(k)})^T d^{(k)} \tag{10.22}$$

em que as constantes μ_1 e μ_2 satisfazem $0 < \mu_1 \leq \mu_2 < 1$. Quanto mais pequeno for μ_1, menor é a redução verificada no valor da função, de iteração para iteração, mais facilmente é verificada a condição (10.21), e no cômputo geral são necessárias mais iterações, uma vez

que o processo progride lentamente. Por sua vez, quanto maior for μ_1, mais restrita é a condição, maior é a redução verificada em f e são necessárias menos iterações para atingir o óptimo.

Acontece, por vezes, que o valor de $\alpha^{(k)}$ aceitável, por forma a originar a redução de f exigida pelas condições, é pequeno demais e o deslocamento resultante $\alpha^{(k)}\, d^{(k)}$ torna-se tão insignificante que o novo ponto 'quase' coincide com $x^{(k)}$. Esta situação ocorre quando o minimizante está muito próximo e o processo torna-se muito lento, obrigando a um número exagerado de cálculos da função f. Sugere-se o uso de $\alpha^{(k)} = 1$ quando esta situação ocorre, mesmo que o valor de f não sofra um decréscimo.

O algoritmo do *critério de Goldstein-Armijo* que usa as condições (10.21) e (10.22) para a aceitação do $\alpha^{(k)}$ é o seguinte:

Algoritmo 10.6 :

1. ler n, k, μ_1, μ_2, $f^{(k)}$ e para $i = 1, \ldots, n$ ler $d_i^{(k)}$, $g_i^{(k)}$ e $x_i^{(k)}$
2. introduzir a função $f(x_1, x_2, \ldots, x_n)$
3. calcular $dd = \sum_{i=1}^{n} g_i^{(k)} d_i^{(k)}$
4. fazer $\alpha = 1$
5. para $j = 1, \ldots, n$ calcular $aux_j = x_j^{(k)} + \alpha\, d_j^{(k)}$, $faux = f(aux_1, aux_2, \ldots, aux_n)$ e $dif = f^{(k)} - faux$
6. se $dif < -\mu_1 \alpha\, dd$ ou $dif > -\mu_2 \alpha\, dd$ então
   - 6.1 fazer $\alpha = \frac{\alpha}{2}$
   - 6.2 calcular $nor = \sqrt{\sum_{i=1}^{n} (d_i^{(k)})^2}$
   - 6.3 se $\alpha\, nor \leq 10^{-8}$ então fazer $\alpha = 1$ e ir para o passo 7, senão ir para o passo 5
7. terminar com $\alpha^{(k)} \leftarrow \alpha$.

Se, em vez do critério de Goldstein-Armijo, se pretender implementar apenas o critério de Armijo, a condição do passo 6. do algoritmo 10.6 deve ser substituída por

6. se $dif < -\mu_1 \alpha\, dd$ então

Se a condição que se quer implementar para a aceitação do valor de $\alpha^{(k)}$, na iteração k, for a indicada no passo 3. do algoritmo genérico dos métodos das direcções de descida, então a condição do passo 6. do algoritmo 10.6 deve ser a seguinte:

6. se $dif \leq 0$ então

Neste caso, apenas se exige uma simples redução no valor da função, de iteração para iteração. O correspondente algoritmo é designado na literatura por algoritmo das repetidas divisões de α por dois.

Quando se implementa esta condição para a aceitação do $\alpha^{(k)}$ não há garantia de que o processo vá convergir para um minimizante local de f.

10.3.8 Método das direcções de descida máxima

Este método usa o vector gradiente $g(x)$ para determinar uma direcção de procura que seja de descida.

Os métodos denominados *do gradiente* baseiam-se na expansão em série de Taylor de $f(x)$, no ponto $x + d$,

$$f(x+d) = f(x) + g(x)^T d + \frac{1}{2} d^T G(x) d + \cdots \tag{10.23}$$

Os termos da série, a partir do segundo, correspondem a uma correcção escalar, que adicionada a $f(x)$, dá o valor de $f(x+d)$. Se a série for truncada logo após o segundo termo, tem-se

$$f(x+d) \approx f(x) + g(x)^T d$$

e o termo $g(x)^T d$, que define a correcção Δf, vem em função das primeiras derivadas de f, definindo uma correcção de *primeira ordem*. O método das direcções de descida máxima é deste tipo.

Se, além do segundo termo, também for usado o terceiro termo, que é função das segundas derivadas através da matriz Hessiana, a correcção diz-se de *segunda ordem*. Em 10.3.10, é apresentado o método de Newton que é caracterizado por definir correcções de segunda ordem.

A correcção ao valor da função, Δf, efeito resultante de uma pequena alteração, d, em x, de primeira ordem, é dada por

$$\Delta f = g(x)^T d = \sum_{i=1}^{n} \frac{\partial f(x)}{\partial x_i} d_i.$$

Como o produto escalar de dois vectores u e v é definido por

$$u^T v = \sum_{i=1}^{n} u_i v_i = \|u\| \|v\| cos(\phi),$$

com ϕ o ângulo definido pelos dois vectores, também $\Delta f = \|g(x)\| \, \|d\| \, cos(\phi)$, depende de $cos(\phi)$, cujo máximo valor positivo é atingido para $\phi = 0$ e o máximo negativo para $\phi = \pi$. Assim, a redução máxima em $f(x)$ ocorre para $\phi = \pi$, o que quer dizer que a alteração d, a verificar-se em x, deve ser feita na direcção negativa do gradiente.

Assim, o vector definido por

$$d = -g(x)$$

define a *direcção de descida máxima*.

Uma vez que estas direcções, d, não apontam necessariamente para o mínimo, o ponto $x + d$ não é já o minimizante procurado e o processo deve ser repetido, agora, a partir de $x + d$.

A procura unidimensional ao longo da direcção de procura, d, do valor do escalar α, que define o comprimento do deslocamento a efectuar ao longo de d, consoante a estratégia utilizada, determina um minimizante local (através de (10.19) e (10.20)), uma boa

aproximação (algoritmo 10.2), ou apenas um novo ponto em que se verifica uma redução significativa no valor de f (algoritmo 10.6). Relembrar o que foi dito em 10.3.7.

Se o valor de α foi determinado exactamente, isto é, se α minimiza f ao longo de d,

$$\frac{d}{d\alpha} f(x + \alpha d) = g(x + \alpha d)^T \, d = 0$$

então as sucessivas direcções de descida máxima definem ângulos rectos e as direcções tendem a tornar-se linearmente dependentes. O método apresenta um comportamento em ziguezague que se traduz num processo muito lento quando está já perto do mínimo. Longe do óptimo este método torna-se mais rápido e a sua convergência, embora sendo linear, é global, isto é, não é necessário que a aproximação inicial esteja próxima do minimizante, para ser possível provar a convergência.

O algoritmo 10.7 apresenta a estrutura algorítmica do método das direcções de descida máxima.

Algoritmo 10.7 :

1. ler n, n_{max} e para $i = 1, \ldots, n$ ler $x_i^{(1)}$
2. introduzir a função $f(x_1, x_2, \ldots, x_n)$
3. introduzir as n primeiras derivadas que definem o vector gradiente:
   $g_1(x_1, x_2, \ldots, x_n)$, $g_2(x_1, x_2, \ldots, x_n)$, ..., $g_n(x_1, x_2, \ldots, x_n)$
4. fazer $k = 1$, calcular $f^{(k)} = f(x_1^{(k)}, x_2^{(k)}, \ldots, x_n^{(k)})$ e para $i = 1, \ldots, n$ calcular $g_i^{(k)} = g_i(x_1^{(k)}, x_2^{(k)}, \ldots, x_n^{(k)})$
5. para $i = 1, \ldots, n$ calcular $d_i = -g_i^{(k)}$
6. calcular o valor de $\alpha^{(k)}$ usando o algoritmo 10.6 ou 10.2
7. para $i = 1, \ldots, n$ calcular $x_i^{(k+1)} = x_i^{(k)} + \alpha^{(k)} d_i$ e $f^{(k+1)} = f(x_1^{(k+1)}, x_2^{(k+1)}, \ldots, x_n^{(k+1)})$
8. para $i = 1, \ldots, n$ calcular $g_i^{(k+1)} = g_i(x_1^{(k+1)}, x_2^{(k+1)}, \ldots, x_n^{(k+1)})$
9. se convergência = .FALSE. (algoritmo 10.3) então
   - 9.1. se $k \geq n_{max}$ então parar, o processo não está a convergir
   - 9.2. fazer $k = k + 1$
   - 9.3. ir para o passo 5
10. terminar com $x^* \leftarrow (x_i^{(k+1)}, i = 1, \ldots, n)$ e $f(x^*) \leftarrow f^{(k+1)}$.

10.3.9 Implementação. Rotina DESMAX

A implementação do algoritmo 10.7 origina a rotina DESMAX. A paragem deste processo iterativo é feita recorrendo-se ao algoritmo 10.3. Um valor aceitável para o parâmetro ε é 10^{-5}.

A rotina DESMAX também recorre ao algoritmo 10.6 para procurar um valor do escalar α, que determine um ponto aceitável, isto é, um novo ponto para o qual a função tenha

um decréscimo significativo. Os valores atribuídos a μ_1 e μ_2 são respectivamente 0.001 e 0.9.

Exemplo 10.12 Na resolução do problema já apresentado nos exemplos 10.8 e 10.10, a rotina DESMAX consegue encontrar a solução ao fim de 2 iterações. Assim, $x^* = (-1.000000, -1.000000)^T$ e $f^* = 0.000000$. O vector inicial do processo foi $(1, 0)^T$.

Exemplo 10.13 Na resolução do problema

$$min_{x \in \mathbf{R}^2} (x_1 + 1)^2 + 100(x_2 + 1)^2$$

a rotina DESMAX converge muito lentamente. Para $\varepsilon = 10^{-4}$, do algoritmo 10.3, obtém-se o vector $(-0.998407, -0.999955)^T$, com $f = 0.000003$, ao fim de 320 iterações. Para se obter uma maior precisão, toma-se $\varepsilon = 10^{-5}$. O processo leva 433 iterações e o valor calculado é $(-0.999873, -1.00004)^T$, com $f = 0.000000$. A solução exacta é o vector $(-1, -1)^T$ atingindo f o valor 0.

10.3.10 Método de Newton

Se a função $f : \mathbf{R}^n \to \mathbf{R}$ tem segundas derivadas parciais contínuas num conjunto aberto e convexo D, que contém um ponto estacionário x^* de f ($g(x^*) = 0$) e se $G(x^*)$ é definida positiva, então o ponto x^* é um minimizante local forte de f em D. Neste caso, um método para determinar os pontos estacionários de f, isto é, um método que resolva o sistema de equações não lineares

$$g(x) = 0\,, \tag{10.24}$$

também é adequado para determinar x^*, o minimizante de f.

Um método eficiente para a resolução numérica do sistema não linear (10.24) é o método de Newton-Raphson (veja-se em 4.2. do Capítulo 4).

A expansão em série de Taylor em (10.23), de 10.3.8, pode também ser usada para aproximar o minimizante x^* da função f. Se o ponto x está perto do minimizante, x^*, tem-se $x^* = x + d$,

$$f(x^*) \approx f(x) + g(x)^T d + \frac{1}{2} d^T G(x) d \tag{10.25}$$

e pretende-se determinar os elementos d_i $(i = 1, 2, ..., n)$ do vector direcção d, por forma a atingir-se o minimizante, a partir de x. Os termos do lado direito de (10.25) definem uma *função quadrática* em d, que aproxima $f(x)$ localmente, na vizinhança de x. O minimizante da quadrática é d.

Para se determinar o vector d, fixando x, deriva-se o lado direito de (10.25) parcialmente em ordem aos elementos d_i,

$$\frac{\partial f(x)}{\partial x_i} + \sum_{j=1}^{n} d_j \frac{\partial^2 f(x)}{\partial x_j \partial x_i}\,, \quad i = 1, ..., n \tag{10.26}$$

e iguala-se a zero, definindo a condição de primeira ordem para o minimizante da quadrática, em termos de x. Na forma matricial, o sistema linear resultante é

$$G(x)d = -g(x)\,. \tag{10.27}$$

Como as derivadas de segunda ordem, de funções quadráticas, são constantes, a aproximação é exacta e o minimizante atinge-se numa só etapa, a partir de um ponto qualquer x.

Quando a função $f(x)$ não é quadrática, o novo ponto obtido $x + d$, com d calculado das equações Newton (10.27) não é provavelmente o minimizante, pelo que o processo deve ser repetido iterativamente. Assim, as equações iterativas do processo são

$$x^{(k+1)} = x^{(k)} + d^{(k)} \tag{10.28}$$

com

$$G(x^{(k)})d^{(k)} = -g(x^{(k)}) \quad \text{para } k = 1, 2, \ldots$$

sendo $x^{(1)}$ uma aproximação inicial à solução. Estas equações definem o *método de Newton*, na sua implementação original e básica.

Teorema 10.1 : Se,

1. $f : \mathbb{R}^n \to \mathbb{R}$ *tem primeiras derivadas parciais contínuas, numa vizinhança aberta e convexa* D *do ponto estacionário* x^* *de* f,
2. $G(x)^{-1}$ *existe e é limitada,* $\|G(x)^{-1}\| \leq w$, *para todo o* $x \in D$
3. $|\partial_i \partial_j \partial_l f(x)| \leq P \quad (i, j, l = 1, ..., n)$, *para todo o* $x \in D$
4. $\Omega \subset D \quad e \quad \frac{wPn^2}{2}\|x - x^*\| \leq r < 1$, *para todo o* $x \in \Omega$
5. $x^{(1)} \in \Omega$

então

i) a sequência $\{x^{(k)}\}$ *gerada pela equação (10.28) converge para* x^* *e*

ii) $\|x^{(k+1)} - x^*\| \leq \frac{wPn^2}{2}\|x^{(k)} - x^*\|^2$, *para* $k \geq 1$.

Deste teorema conclui-se que se $x^{(1)}$ está suficientemente perto de x^* e as outras condições também são satisfeitas, então a sequência de pontos gerada pelo método de Newton converge para x^* e a convergência é quadrática. Por vezes, não é possível obter-se uma boa estimativa inicial $x^{(1)}$ e o método pode não convergir.

Na prática, existem outras razões que levam o método de Newton, implementado de acordo com as equações em (10.28), a falhar a convergência para a solução desejada.

Enumeram-se, de seguida, as situações que podem ocorrer e que fazem com que o método de Newton possa não convergir.

1. A matriz $G(x^{(k)})$ é definida positiva sendo o vector direcção, $d^{(k)}$, calculado da equação (10.27) de descida. No entanto, o comprimento da direcção é tal que $f(x^{(k+1)}) > f(x^{(k)})$, e o novo ponto não é melhor do que o anterior, no sentido de que a função não foi reduzida.

2. O vector direcção $d^{(k)}$ calculado da equação (10.27) é quase ortogonal a $g(x^{(k)})$

$$g(x^{(k)})^T d^{(k)} \simeq 0.$$

Neste caso, não é possível progredir ao longo de $d^{(k)}$.

3. A direcção Newton não é de descida para f,

$$g(x^{(k)})^T d^{(k)} > 0.$$

Nesta situação, não é possível garantir a existência de um valor positivo para α que verifique $f(x^{(k)} + \alpha d^{(k)}) < f(x^{(k)})$.

4. A matriz $G(x^{(k)})$ é singular. A direcção de (10.27) pode não ser sequer definida.

Para cada uma das situações referidas, apresentam-se possíveis soluções para ultrapassar esses inconvenientes.

Quando ocorre o caso 1., é possível garantir a existência de um valor positivo para $\alpha^{(k)}$, de tal forma que, pelo menos, $f(x^{(k)} + \alpha^{(k)} d^{(k)}) < f(x^{(k)})$.

Assim, no cálculo de $\alpha^{(k)}$, pode usar-se a procura unidimensional exacta, ou um critério baseado na condição de Armijo ou ainda na condição de Goldstein-Armijo (algoritmo 10.6).

Se aparecer $d^{(k)}$ quase ortogonal a $g(x^{(k)})$ (caso 2.), a direcção deve ser substituída por outra, nomeadamente pela direcção de descida máxima, $-g(x^{(k)})$.

Como esta direcção é de descida para f, garante-se a existência de $\alpha^{(k)}$. A implementação do algoritmo 10.6 completa a iteração.

Na prática e devido aos erros de arredondamento, considera-se que existe ortogonalidade entre $d^{(k)}$ e $g(x^{(k)})$ se

$$|g(x^{(k)})^T d^{(k)}| \leq \eta \|g(x^{(k)})\| \|d^{(k)}\|$$

com η uma quantidade positiva e próxima de zero.

No caso 3., como a direcção $d^{(k)}$ não aponta necessariamente no sentido descendente de f, facto que é verificado na prática através de

$$g(x^{(k)})^T d^{(k)} > \eta \|g(x^{(k)})\| \|d^{(k)}\|,$$

usa-se $-d^{(k)}$ que já define uma direcção de descida para f, garantindo-se assim a existência de $\alpha^{(k)}$. A implementação do algoritmo 10.6 completa a iteração.

Finalmente, se $G(x^{(k)})$ é singular (caso 4.), situação que é detectada durante o processo de eliminação de Gauss com pivotagem parcial, o sistema Newton pode não ter solução.

É possível ultrapassar esta dificuldade, atribuindo ao vector $d^{(k)}$ a direcção de descida máxima, $-g(x^{(k)})$. Mais uma vez, a implementação do algoritmo 10.6 completa a iteração.

A introdução destes esquemas de segurança no processo iterativo definido pelas equações em (10.28), origina um método, conhecido por *método de segurança de Newton*, cujo algoritmo é apresentado de seguida.

Algoritmo 10.8 :

1. ler n, n_{max}, η e para $i = 1, \ldots, n$ ler $x_i^{(1)}$
2. introduzir a função $f(x_1, x_2, \ldots, x_n)$
3. introduzir as n primeiras derivadas que definem o vector gradiente:
   $g_1(x_1, x_2, \ldots, x_n)$, $g_2(x_1, x_2, \ldots, x_n)$, $\ldots$, $g_n(x_1, x_2, \ldots, x_n)$
4. introduzir as 1/2 n(n+1) segundas derivadas que definem a matriz Hessiana:
   $G_{11}(x_1, x_2, \ldots, x_n), \ldots, G_{1n}(x_1, x_2, \ldots, x_n), G_{22}(x_1, x_2, \ldots, x_n), \ldots, G_{2n}(x_1, x_2, \ldots, x_n)$,
   $G_{33}(x_1, x_2, \ldots, x_n), \ldots, G_{3n}(x_1, x_2, \ldots, x_n), \ldots, G_{n-1\,n-1}(x_1, x_2, \ldots, x_n)$,
   $G_{n-1\,n}(x_1, x_2, \ldots, x_n), G_{n\,n}(x_1, x_2, \ldots, x_n)$
5. fazer $k = 1$, calcular $f^{(k)} = f(x_1^{(k)}, x_2^{(k)}, \ldots, x_n^{(k)})$ e para $i = 1, \ldots, n$ calcular
   $g_i^{(k)} = g_i(x_1^{(k)}, x_2^{(k)}, \ldots, x_n^{(k)})$
6. para $i = 1, \ldots, n$ e $j = i, \ldots, n$
   - 6.1. calcular $G_{i\,j} = G_{i\,j}(x_1^{(k)}, x_2^{(k)}, \ldots, x_n^{(k)})$
   - 6.2. se $i \neq j$ então fazer $G_{j\,i} = G_{i\,j}$
7. formar o sistema das n equações lineares $Gd = -g^{(k)}$ e resolver (algoritmo 3.3)
   - 7.1. se o sistema tem solução única então colocar a solução em $d_1, d_2, \ldots, d_n$
     - 7.1.1. calcular $nor1 = \sqrt{\sum_{i=1}^{n} (g_i^{(k)})^2}$, $nor2 = \sqrt{\sum_{i=1}^{n} d_i^2}$ e $dd = \sum_{i=1}^{n} g_i^{(k)} d_i$
     - 7.1.2. se $|dd| \leq \eta\, nor1\, nor2$ então para $i = 1, \ldots, n$ fazer $d_i = -g_i^{(k)}$
     - 7.1.3. senão, se $dd > \eta\, nor1\, nor2$ então para $i = 1, \ldots, n$ fazer $d_i = -d_i$
   - 7.2. senão para $i = 1, \ldots, n$ fazer $d_i = -g_i^{(k)}$
8. calcular o valor de $\alpha^{(k)}$ usando o algoritmo 10.6 (ou o algoritmo 10.2)
9. para $i = 1, \ldots, n$ calcular $x_i^{(k+1)} = x_i^{(k)} + \alpha^{(k)} d_i$ e $f^{(k+1)} = f(x_1^{(k+1)}, \ldots, x_n^{(k+1)})$
10. para $i = 1, \ldots, n$ calcular $g_i^{(k+1)} = g_i(x_1^{(k+1)}, x_2^{(k+1)}, \ldots, x_n^{(k+1)})$
11. se convergência = .FALSE. (algoritmo 10.3) então
    - 11.1. se $k \geq n_{max}$ então parar, o processo não está a convergir
    - 11.2. fazer $k = k + 1$
    - 11.3. ir para o passo 6
12. terminar com $x^* \leftarrow (x_i^{(k+1)}, i = 1, \ldots, n)$ e $f(x^*) \leftarrow f^{(k+1)}$.

Além do método de Newton básico e do método de segurança de Newton, existem ainda outros métodos de segunda ordem. São exemplos, os métodos de Greenstadt e de Marquardt-Levenberg. Embora cada um deles tenha as suas vantagens, todos apresentam uma dificuldade comum - a necessidade de calcular as segundas derivadas de f para formar a matriz Hessiana. No entanto, este imperativo pode acabar por mostrar-se vantajoso, uma vez que estes métodos são, em geral, mais rápidos e eficientes, em termos do número de iterações e de cálculos de $f(x)$ quando comparados com os métodos de primeira ordem. Além disso, com a informação das segundas derivadas é possível assegurar as condições suficientes para um minimizante.

10.3.11 Implementação. Rotina NEWSEG

A rotina NEWSEG implementa o algoritmo 10.8 do método de segurança de Newton. A paragem do processo é feita pelo critério apresentado no algoritmo 10.3. A ε é dado o valor 10^{-5}. A rotina utiliza o algoritmo 10.6, para a escolha do α, com $\mu_1 = 0.001$ e $\mu_2 = 0.9$. A η é dado o valor 10^{-10}.

Exemplo 10.14 A partir do vector inicial $(1,0)^T$, a rotina NEWSEG calcula a solução $(-1.000000, -1.000000)^T$, com $f^* = 0.000000$, do problema já apresentado em 10.8, 10.10 e 10.12, em apenas 2 iterações.

Exemplo 10.15 Para resolver o problema

$$min_{x \in \mathbf{R}^2} f_1(x_1, x_2) = x_1^2 + x_2^2 - \frac{1}{3} x_1 x_2$$

já apresentado nos exemplos 10.9 e 10.11, a rotina NEWSEG leva 2 iterações para atingir o vector $(0.000000, 0.000000)^T$, com $f_1^* = 0.000000$. O vector inicial é $(1,1)^T$.

O problema

$$min_{x \in \mathbf{R}^2} f_2(x_1, x_2) = 144 x_1^2 + 4 x_2^2 - 8 x_1 x_2$$

é equivalente ao primeiro, no sentido de que tem a mesma solução, no entanto, não está bem escalonado. Isto é, qualquer variação de uma unidade nas variáveis x_1 e x_2 originam variações em f_2 muito distintas. É possível escalonar o problema, por forma a torná-lo bem escalonado e fácil de resolver. Se se fizer uma mudança de variáveis, de tal forma que $x_1 \leftarrow \frac{x_1}{12}$ e $x_2 \leftarrow \frac{x_2}{2}$ então

$$f_2(\frac{x_1}{12}, \frac{x_2}{2}) = f_1(x_1, x_2)$$

sendo f_1 a função original deste exemplo.

O problema $min_{x \in \mathbf{R}^2} f_2(x_1, x_2)$ é resolvido ao fim de 2 iterações e obtém-se a solução $(0,0)^T$, com $f^* = 0$. O vector inicial foi também $(1,1)^T$. Embora, em geral, este problema seja mais difícil de resolver, as duas implementações de NEWSEG foram igualmente eficientes.

10.3.12 Métodos do tipo Quasi - Newton

O método de Newton precisa do cálculo das segundas derivadas na forma da matriz Hessiana. Estas derivadas podem ser difíceis de calcular quando a função $f(x)$ tem uma forma complicada. Além disso, dão trabalho a calcular quando n é grande, pois, no total, são

$n(n+1)/2$ segundas derivadas diferentes. Assim, nestas situações, parece ser aconselhável evitar o cálculo de $G(x)$. Uma estratégia possível consiste em

1. obter uma estimativa da matriz Hessiana na primeira iteração, que vai ser denotada por $B^{(1)}$, e com ela calcular o vector direcção $d^{(1)}$, através da resolução do sistema de equações lineares (relembrar o sistema Newton (10.27))

$$B^{(1)}d^{(1)} = -g(x^{(1)}).$$

   A nova aproximação calculada passa a ser: $x^{(2)} = x^{(1)} + \alpha^{(1)}d^{(1)}$.

2. Nas iterações seguintes, $k = 2, \ldots$, actualiza-se a matriz $B^{(k)}$, usando um esquema de actualização simples, apenas baseado na informação das primeiras derivadas. A respectiva direcção de procura, $d^{(k)}$, é calculada através da resolução do sistema

$$B^{(k)}d^{(k)} = -g(x^{(k)}).$$

É ainda possível evitar a resolução do sistema de equações lineares, em todas as iterações, optando por esquemas de actualização da matriz que aproxima a inversa da Hessiana, em vez de aproximar a Hessiana. Essas matrizes vão ser denotadas, daqui para a frente, por $H^{(k)}$. Para o cálculo da direcção $d^{(k)}$ apenas se efectua o produto da matriz pelo vector: $d^{(k)} = -H^{(k)}g(x^{(k)})$.

As condições que podem ser usadas para definir fórmulas de actualização, quer para aproximar a Hessiana, quer para aproximar a sua inversa, são as seguintes.

A matriz $B^{(k)}$ deve estimar o melhor possível $G(x^{(k)})$, isto é, deve simular a curvatura de $f(x)$ no ponto $x^{(k)}$,

$$G(x^{(k)})s^{(k)} = y^{(k)} \tag{10.29}$$

em que

$$s^{(k)} = \alpha^{(k)}d^{(k)} = x^{(k+1)} - x^{(k)},$$

e

$$y^{(k)} = g(x^{(k+1)}) - g(x^{(k)})$$

é a variação verificada no vector gradiente, quando se passa do ponto $x^{(k)}$ para $x^{(k+1)}$.

Para calcular $y^{(k)}$ é necessário conhecer $x^{(k+1)}$, que por sua vez, é obtido a partir de $B^{(k)}$. Assim, precisa-se de $B^{(k)}$ antes de calcular o ponto da iteração seguinte. Logo deve ser a matriz $B^{(k+1)}$ a verificar a equação (10.29),

$$B^{(k+1)}s^{(k)} = y^{(k)} \tag{10.30}$$

e que é conhecida por *condição Quasi-Newton ou secante*. A correspondente condição, para a aproximação à inversa da Hessiana, $H^{(k+1)}$, é,

$$H^{(k+1)}y^{(k)} = s^{(k)}. \tag{10.31}$$

Para actualizar $B^{(k)}$ usa-se a matriz $C^{(k)}$, verificando-se sempre

$$B^{(k+1)} = B^{(k)} + C^{(k)}, \ \ k = 1, \ldots$$

e, da mesma forma, $E^{(k)}$ é a matriz que actualiza a $H^{(k)}$, verificando-se

$$H^{(k+1)} = H^{(k)} + E^{(k)}, \ \ k = 1, \ldots$$

Existem várias escolhas possíveis para as matrizes $C^{(k)}$, ou $E^{(k)}$. Podem ser seleccionadas por forma a serem:

i) matrizes de característica um,

ii) matrizes de característica dois, ou

iii) matrizes de norma mínima.

Uma vez que o critério iii) origina as mesmas matrizes do que os dois primeiros, apenas são referidas fórmulas que satisfazem i) ou ii).

Para evitar a resolução de sistemas de equações, apenas são apresentadas fórmulas de actualização de $H^{(k)}$, como estimativa da inversa da Hessiana. Os métodos resultantes da aproximação da inversa da Hessiana por fórmulas de actualização, e posterior determinação da direcção de procura, através de

$$d^{(k)} = -H^{(k)} g(x^{(k)}) \tag{10.32}$$

chamam-se *métodos do tipo Quasi-Newton.*

Para que o método seja eficiente, espera-se que $H^{(k)}$ vá convergir para $G(x^*)^{-1}$, à medida que k aumenta, isto é, à medida que se vão fazendo iterações. No entanto, esta condição não é necessária para que o método convirja.

A fórmula mais conhecida para actualizar $H^{(k)}$ é devida a Davidon[10], Fletcher e Powell (D.F.P.),

$$H_{DFP}^{(k+1)} = H^{(k)} - \frac{H^{(k)} y^{(k)} y^{(k)T} H^{(k)}}{y^{(k)T} H^{(k)} y^{(k)}} + \frac{s^{(k)} s^{(k)T}}{s^{(k)T} y^{(k)}} \tag{10.33}$$

em que $s^{(k)} = \alpha^{(k)} d^{(k)}$ e $\alpha^{(k)}$ é calculado por uma técnica de procura unidimensional.

A matriz inicial $H^{(1)}$ pode ser arbitrária, desde que seja definida positiva e simétrica, pois algumas fórmulas de actualização conservam estas propriedades de iteração para iteração. A fórmula D.F.P. é uma delas.

[10]Um dos trabalhos mais citados na literatura sobre Optimização é da autoria de W.C. Davidon e data de 1959. O trabalho só recentemente foi publicado num Jornal - SIAM Journal on Optimization, 1991 - e tem como título 'Variable Metric Method for Minimization'. Hoje em dia, este tipo de métodos é conhecido por método Quasi-Newton. Davidon é um físico matemático que recebeu o Ph.D. em Chicago, em 1954, presentemente lecciona em Haverford no Departamento de Matemática, embora tenha sido professor de Física até 1981 (SIAM NEWS, 1990).

Uma matriz definida positiva em (10.32) origina uma direcção de descida e este facto é vantajoso quando se pretende que a função decresça significativamente de iteração para iteração. O algoritmo correspondente diz-se então *estável*.

Na maior parte dos casos toma-se $H^{(1)} = I$, resultando $d^{(1)}$ como direcção de descida máxima.

Esta fórmula pertence ao caso ii) e pode-se provar que $s^{(1)}, s^{(2)}, ..., s^{(n)}$ formam um conjunto de direcções conjugadas e a inversa de $H^{(n)}$ coincide com G, constante, quando a função objectivo é quadrática da forma $f(x) = \frac{1}{2}x^T Gx + c^T x$.

Quando a função $f(x)$ é quadrática e tem um mínimo, ela é convexa. A matriz G é simétrica e semidefinida positiva. Se, além disso, a característica de G for n, então ela é definida positiva. Nestas condições, um conjunto de n vectores $u_1, u_2, \ldots, u_n$ diz-se *mutuamente conjugado* em relação a G, se $u_i^T G u_j = 0$ para $i \neq j$.

Um algoritmo, para o qual, é possível provar que minimiza uma função quadrática e convexa num número finito de iterações, menor ou igual a n, diz-se que exibe a *propriedade da terminação quadrática*. Esta propriedade é desejável embora as suas implicações práticas não sejam ainda muito claras quando se resolvem problemas com funções não quadráticas. No entanto, todas as funções contínuas e continuamente diferenciáveis são quadráticas na vizinhança de um minimizante isolado.

Uma fórmula de actualização dos $H^{(k)}$, que gera matrizes simétricas e que pertence à classe i), é

$$H_D^{(k+1)} = H^{(k)} + \frac{(s^{(k)} - H^{(k)}y^{(k)})(s^{(k)} - H^{(k)}y^{(k)})^T}{(s^{(k)T}y^{(k)} - y^{(k)T}H^{(k)}y^{(k)})} \tag{10.34}$$

devida a Davidon.

A utilização desta fórmula não exige um processo de procura unidimensional exacta e gera direcções conjugadas quando a função objectivo é quadrática. Tal como a fórmula de D.F.P., a de Davidon também exibe a propriedade da terminação quadrática.

A fórmula de Davidon não gera um algoritmo estável. No entanto, é possível ultrapassar este inconveniente introduzindo um esquema que garanta que as direcções de procura usadas apontem no sentido descendente de $f(x)$. Este esquema define a *técnica do recomeço* e consiste em repôr, em $H^{(k)}$, a matriz identidade, que é simétrica e definida positiva. Nessas iterações, a direcção de procura passa a ser de descida.

Outra fórmula de característica dois, proposta por Broyden, Fletcher, Goldfarb e Shanno e conhecida por B.F.G.S., é

$$H_{BFGS}^{(k+1)} = (I - \frac{s^{(k)}y^{(k)T}}{s^{(k)T}y^{(k)}})\, H^{(k)}\, (I - \frac{y^{(k)}s^{(k)T}}{y^{(k)T}s^{(k)}}) + \frac{s^{(k)}s^{(k)T}}{s^{(k)T}y^{(k)}} \tag{10.35}$$

que também satisfaz a propriedade da terminação quadrática.

É possível definir uma classe de fórmulas de actualização simétricas, que satisfazem a propriedade $H^{(k)}y^{(j)} = s^{(j)}, j = 1, \ldots, k-1$ e que utilizam procura unidimensional ao longo de $d^{(k)}$, a partir de

$$H^{(k)} = (1-\gamma)H_{DFP}^{(k)} + \gamma H_{BFGS}^{(k)}, \quad 0 \leq \gamma \leq 1\,. \tag{10.36}$$

As fórmulas (10.33), (10.35) e (10.36) geram matrizes definidas positivas se for introduzida a procura unidimensional exacta no algoritmo. Uma condição necessária e suficiente para que as fórmulas de D.F.P. e B.F.G.S. gerem matrizes definidas positivas é

$$s^{(k)T} y^{(k)} > 0. \tag{10.37}$$

Pelo facto das matrizes $H^{(k)}$ serem definidas positivas, e mesmo que f seja uma função não quadrática, a direcção de procura nunca é ortogonal ao gradiente,

$$-g(x^{(k)})^T H^{(k)} g(x^{(k)}) \neq 0,$$

e os denominadores dos termos de actualização em (10.33) ((10.35) e (10.36)) teoricamente não se anulam e obtém-se um decréscimo de f em cada iteração. O método é, assim, estável.

Na prática, quando f não é unimodal ao longo da direcção de procura e o algoritmo de procura unidimensional gera, apenas, um mínimo local, pode surgir um aumento do valor da função.

Os erros de arredondamento que se cometem ao longo do processo de actualização podem originar matrizes $H^{(k)}$ não definidas positivas e o método acaba por divergir.

Além disso, a sequência

$$-g(x^{(k)})^T H^{(k)} g(x^{(k)})$$

é monótona decrescente e, se $g(x^{(k)})$ toma valores pequenos (perto de pontos sela ou outros pontos estacionários), também ela atinge valores próximos de zero. Esta situação também ocorre quando a função é exageradamente não simétrica (mal escalonada) e a matriz Hessiana tem valores próprios grandes. A correspondente matriz H torna-se quase singular.

Como consequência, os denominadores dos termos de actualização, da matriz $H^{(k)}$, aproximam-se de zero e uma das seguintes situações pode ocorrer:

- 'overflow' na actualização de $H^{(k)}$;
- um deslocamento ao longo da direcção de procura muito longo devido à quase ortogonalidade da direcção com o gradiente.

Estes problemas podem ser ultrapassados com a introdução da técnica do recomeço. Assim, de n em n iterações (outros autores têm proposto de $n+1$ em $n+1$ iterações) a matriz $H^{(k)}$ torna a ser a identidade.

A versão algorítmica B.F.G.S. do método Quasi-Newton, com a técnica do recomeço implementada de n em n iterações, é a que a seguir se apresenta:

Algoritmo 10.9 :

1. `ler` n, n_{max} `e para` $i = 1, \ldots, n$ `ler` $x_i^{(1)}$
2. `introduzir a função` $f(x_1, x_2, \ldots, x_n)$

3. introduzir as n primeiras derivadas que definem o vector gradiente:
   $g_1(x_1, x_2, \ldots, x_n)$, $g_2(x_1, x_2, \ldots, x_n)$, ..., $g_n(x_1, x_2, \ldots, x_n)$
4. fazer $k = 0$, calcular $f^{(1)} = f(x_1^{(1)}, x_2^{(1)}, \ldots, x_n^{(1)})$ e para $i = 1, \ldots, n$ calcular $g_i^{(1)} = g_i(x_1^{(1)}, x_2^{(1)}, \ldots, x_n^{(1)})$
5. para $i = 1, \ldots, n$ e $j = i, \ldots, n$
   5.1. se $i = j$ então fazer $H_{ij} = 1$
   5.2. senão fazer $H_{ji} = H_{ij} = 0$
6. fazer $k = k + 1$ e para $i = 1, \ldots, n$ calcular $d_i = -\sum_{j=1}^{n} H_{ij} g_j^{(k)}$
7. calcular o valor de $\alpha^{(k)}$ usando o algoritmo 10.6 ou 10.2
8. para $i = 1, \ldots, n$ calcular $x_i^{(k+1)} = x_i^{(k)} + \alpha^{(k)} d_i$ e $f^{(k+1)} = f(x_1^{(k+1)}, x_2^{(k+1)}, \ldots, x_n^{(k+1)})$
9. para $i = 1, \ldots, n$ calcular $g_i^{(k+1)} = g_i(x_1^{(k+1)}, x_2^{(k+1)}, \ldots, x_n^{(k+1)})$
10. se convergência = .FALSE. (algoritmo 10.3) então
    10.1. se $k \geq n_{max}$ então parar, o processo não está a convergir
    10.2. se quer recomeçar e $k =$ múltiplo de n então ir para o passo 5
    10.3. senão actualizar a matriz H (B.F.G.S.) :
      10.3.1. calcular os vectores y e s: para $i = 1, \ldots, n$ calcular $y_i = g_i^{(k+1)} - g_i^{(k)}$ e $s_i = \alpha^{(k)} d_i$
      10.3.2. calcular $den = \sum_{i=1}^{n} s_i y_i$
      10.3.3. para $i = 1, \ldots, n$ e $j = 1, \ldots, n$
        10.3.3.1. calcular $mat_{ij} = \frac{s_i y_j}{den}$
        10.3.3.2. se $i = j$ então fazer $mat_{ij} = 1 - mat_{ij}$ senão fazer $mat_{ij} = -mat_{ij}$
        10.3.3.3. fazer $aux_{ij} = H_{ij}$
      10.3.4. para $i = 1, \ldots, n$ e $j = 1, \ldots, n$ calcular $H_{ij} = \sum_{l=1}^{n} mat_{il} aux_{lj}$
      10.3.5. para $i = 1, \ldots, n$ e $j = 1, \ldots, n$ calcular $aux_{ij} = \sum_{l=1}^{n} H_{il} mat_{jl}$
      10.3.6. para $i = 1, \ldots, n$ e $j = i, \ldots, n$
        10.3.6.1. calcular $mat_{ij} = \frac{s_i s_j}{den}$
        10.3.6.2. se $i \neq j$ então fazer $mat_{ji} = mat_{ij}$
      10.3.7. para $i = 1, \ldots, n$ e $j = 1, \ldots, n$ calcular $H_{ij} = aux_{ij} + mat_{ij}$
      10.3.8. ir para o passo 6
11. terminar com $x^* \leftarrow (x_i^{(k+1)}, i = 1, \ldots, n)$ e $f(x^*) \leftarrow f^{(k+1)}$.

10.3.13 Implementação. Rotina QUANEW

A implementação do algoritmo 10.9 origina a rotina QUANEW.

Exemplo 10.16 A rotina QUANEW é bastante eficiente na resolução do problema

$$min_{x\in\mathbf{R}^2}\ (x_1+1)^2+(x_2+1)^2,$$

que já surgiu nos exemplos 10.8, 10.10, 10.12 e 10.14. Os parâmetros de entrada são idênticos aos usados nos casos 10.12 e 10.14 e ao fim de 2 iterações obtém-se $(-1.000000, -1.000000)^T$, com $f^* = 0.000000$.

Exemplo 10.17 Duas implementações com características distintas surgem na resolução de

$$min_{x\in\mathbf{R}^2}\ f_1(x_1,x_2)$$

e

$$min_{x\in\mathbf{R}^2}\ f_2(x_1,x_2)$$

com f_1 e f_2 definidas no exemplo 10.15. Para o primeiro problema, bem escalonado, QUANEW converge ao fim de 3 iterações, para a solução $(0,0)^T$ e para o segundo problema, mal escalonado, converge muito lentamente.

10.3.14 Método dos gradientes conjugados

Nos problemas considerados de grandes dimensões, em que o número de variáveis é elevado, interessa evitar o cálculo das segundas derivadas de $f(x)$ e até a resolução do sistema de equações lineares para o cálculo da direcção de procura é desaconselhável. Assim, os métodos baseados nas equações Newton (veja-se em 10.3.10) e os do tipo Quasi-Newton (em 10.3.12) não devem ser implementados.

O método das direcções de descida máxima, calculadas apenas em função do vector gradiente (em 10.3.8), pode ser uma alternativa, no entanto, tem tendência a gerar direcções $d^{(k)}$ linearmente dependentes, originando um processo iterativo muito lento.

Para evitar a dependência linear, deve assegurar-se que as direcções $d^{(k)}$ sejam conjugadas, isto é, verifiquem

$$d^{(k)T}Gd^{(j)}=0,\quad k\neq j,$$

em que G é a matriz Hessiana, constante, da função quadrática,

$$f(x)=c^Tx+\frac{1}{2}x^TGx. \tag{10.38}$$

Usando procura unidimensional exacta ao longo de cada uma destas direcções, obtém-se a solução ao fim de n (ou menos do que n) iterações. Tem-se, assim, a propriedade da terminação quadrática. Da definição (10.29) tem-se

$$y^{(k)}=g(x^{(k+1)})-g(x^{(k)})=G(x^{(k+1)}-x^{(k)})=\alpha^{(k)}Gd^{(k)}$$

e a condição $d^{(k)T}Gd^{(j)}=0,\ \ k\neq j$, é equivalente à condição de ortogonalidade $y^{(k)T}d^{(j)}=0$.

Consegue-se um conjunto de direcções mutuamente conjugadas, tomando $d^{(1)} = -g(x^{(1)})$ e, para as direcções subsequentes $k = 2, 3, ...$, considerando $d^{(k)}$ como uma combinação linear de $g(x^{(k)})$ e das $k-1$ direcções anteriores,

$$d^{(k)} = -g(x^{(k)}) + \sum_{i=1}^{k-1} \beta_k^{(i)} d^{(i)}. \tag{10.39}$$

Quando a direcção $d^{(k)}$ é definida por (10.39), $g(x^{(k)})$ torna-se uma combinação linear dos $d^{(1)}, d^{(2)}, \ldots, d^{(k)}$.

Quando cada $x^{(j)}$ for obtido por procura unidimensional exacta, ao longo da direcção $d^{(j-1)}$, verifica-se para $j = 2, \ldots, k$

$$g(x^{(j)})^T d^{(i)} = 0, \quad j > i$$

e

$$g(x^{(k)})^T g(x^{(i)}) = 0, \quad i < k.$$

A direcção $d^{(k)}$ pode ser calculada por forma a ser conjugada com $d^{(1)}, d^{(2)}, \ldots, d^{(k-1)}$. Multiplicando (10.39), à esquerda, por $d^{(i)T} G$, e usando a definição de direcções conjugadas e (10.29), tem-se

$$d^{(i)T} G d^{(k)} = -d^{(i)T} G g(x^{(k)}) + \sum_{j=1}^{k-1} \beta_k^{(j)} d^{(i)T} G d^{(j)}$$

ou

$$d^{(i)T} G d^{(k)} = -\frac{1}{\alpha^{(k)}} (g(x^{(i+1)}) - g(x^{(i)}))^T g(x^{(k)}) + \beta_k^{(i)} d^{(i)T} G d^{(i)}. \tag{10.40}$$

O primeiro termo do lado direito, desta igualdade, anula-se, sempre que $i < k-1$. Assim, para que $d^{(k)}$ seja conjugado com $d^{(i)}$, para $i < k-1$, escolhem-se os $\beta_k^{(i)}$ iguais a zero. Como o único coeficiente que fica diferente de zero é o $\beta_k^{(k-1)}$, a partir de agora será identificado com um único índice: $\beta^{(k-1)}$.

Para se calcular o valor de $\beta^{(k-1)}$ que gera $d^{(k)}$ como uma direcção conjugada com $d^{(k-1)}$, multiplica-se (10.39), à esquerda, por $y^{(k-1)T}$, que dá

$$0 = -y^{(k-1)T} g(x^{(k)}) + \beta^{(k-1)} y^{(k-1)T} d^{(k-1)}$$

devido à condição de ortogonalidade, ou seja

$$\beta^{(k-1)} = \frac{y^{(k-1)T} g(x^{(k)})}{y^{(k-1)T} d^{(k-1)}}. \tag{10.41}$$

Esta fórmula é devida a Hestenes[11] e Stiefel[12] (H.S.).

Assim, o *método dos gradientes conjugados*[13] gera uma sequência de aproximações à solução

$$x^{(k+1)} = x^{(k)} + \alpha^{(k)} d^{(k)}$$

em que a direcção, da iteração k, é calculada através de

$$d^{(k)} = -g(x^{(k)}) + \beta^{(k-1)} d^{(k-1)} \tag{10.42}$$

com $\beta^{(k-1)}$ calculado de (10.41) e $y^{(k-1)} = g(x^{(k)}) - g(x^{(k-1)})$.

Existem fórmulas alternativas para o cálculo do escalar $\beta^{(k-1)}$.

Se a procura unidimensional exacta tiver sido implementada ao longo das direcções $d^{(k)}$ e $d^{(k-1)}$, verificam-se as seguintes condições:

a) $d^{(k-1)T} g(x^{(k)}) = 0$

b) $d^{(k-2)T} g(x^{(k-1)}) = 0$

e a fórmula de H.S. simplifica e origina a de Polak e Ribiére (P.R.)

$$\beta^{(k-1)} = \frac{y^{(k-1)T} g(x^{(k)})}{g(x^{(k-1)})^T g(x^{(k-1)})} . \tag{10.43}$$

Se, além disso, a função $f(x)$ for quadrática, tem-se

c) $g(x^{(k)})^T g(x^{(k-1)}) = 0$

e a fórmula (10.43) simplifica, originando a de Fletcher e Reeves[14](F.R.)

$$\beta^{(k-1)} = \frac{g(x^{(k)})^T g(x^{(k)})}{g(x^{(k-1)})^T g(x^{(k-1)})} . \tag{10.44}$$

[11]M.R. Hestenes (1906-1991) nasceu nos Estados Unidos. Recebeu o seu B.A. em 1927, o M.A. em 1928 pela Universidade de Wisconsin e o Ph.D. em 1932 pela Universidade de Chicago. Esteve na Faculdade de Chicago como Professor Assistente, em 1937, e como Professor Associado até 1947. Foi então ocupar um lugar de Professor no UCLA, onde esteve até se reformar em 1973. Desenvolveu outras funções não académicas em várias instituições e foi Vice-presidente da 'American Mathematical Society'. A contribuição mais importante do seu trabalho, desenvolvido entre 1936 e 1951, está relacionada com o conceito de direcções conjugadas e que deu origem ao método dos gradientes conjugados (SIAM NEWS, V24, N. 2, 1991).

[12]E. Stiefel desenvolveu independentemente de Hestenes o método dos gradientes conjugados, em Zurique, 1951. A primeira apresentação pública dos trabalhos parece ter sido feita no 'Symposium on Simultaneous Linear Equations and the Determination of Eigenvalues' em Agosto de 1951. A publicação conjunta dos trabalhos foi feita um ano mais tarde (SIAM NEWS, V24, N. 2, (1991) e Golub e O'Leary (1989)).

[13]O método dos gradientes conjugados para a resolução de um sistema de equações lineares representou uma inovação muito importante, nos princípios dos anos 60 e começou a ser bastante usado nos meados dos anos 70. Posteriormente foi estendido à resolução de sistemas de equações não lineares e aos problemas de optimização (Golub e O'Leary (1989)).

[14]R. Fletcher e C.M. Reeves num trabalho publicado no 'Computer Journal' de 1964, generalizam os gradientes conjugados a funções não quadráticas, implementando um algoritmo de procura unidimensional (Golub e O'Leary (1989)).

As diferentes definições de $\beta^{(k-1)}$ afectam o número de vectores que é necessário armazenar durante o processo.

Quando $f(x)$ é quadrática, atinge-se o mínimo em n (ou menos do que n) iterações. No entanto, se $f(x)$ não é quadrática, o processo iterativo leva, em geral, mais do que n iterações. Para simular um comportamento semelhante ao de um problema quadrático é vantajoso introduzir a *técnica do recomeço*, repondo em $d^{(k)}$ a direcção de descida máxima, $-g(x^{(k)})$, de n em n iterações.

O algoritmo 10.10 que apresenta o método dos gradientes conjugados, é baseado na fórmula de Hesteness e Stiefel para calcular o escalar β e implementa a técnica do recomeço.

Algoritmo 10.10 :

1. ler n, n_{max} e para $i = 1, \ldots, n$ ler $x_i^{(1)}$
2. introduzir a função $f(x_1, x_2, \ldots, x_n)$
3. introduzir as n primeiras derivadas que definem o vector gradiente:
   $g_1(x_1, x_2, \ldots, x_n)$, $g_2(x_1, x_2, \ldots, x_n)$, ..., $g_n(x_1, x_2, \ldots, x_n)$
4. fazer $k = 0$, calcular $f^{(1)} = f(x_1^{(1)}, x_2^{(1)}, \ldots, x_n^{(1)})$ e para $i = 1, \ldots, n$ calcular
   $g_i^{(1)} = g_i(x_1^{(1)}, x_2^{(1)}, \ldots, x_n^{(1)})$
5. fazer $\beta = 0$ e para $i = 1, \ldots, n$ fazer $d_i^{(k)} = 0$
6. fazer $k = k + 1$ e para $i = 1, \ldots, n$ calcular $d_i^{(k)} = -g_i^{(k)} + \beta\, d_i^{(k-1)}$
7. calcular o valor de $\alpha^{(k)}$ usando o algoritmo 10.6 ou 10.2
8. para $i = 1, \ldots, n$ calcular $x_i^{(k+1)} = x_i^{(k)} + \alpha^{(k)} d_i^{(k)}$ e $f^{(k+1)} = f(x_1^{(k+1)}, x_2^{(k+1)}, \ldots, x_n^{(k+1)})$
9. para $i = 1, \ldots, n$ calcular $g_i^{(k+1)} = g_i(x_1^{(k+1)}, x_2^{(k+1)}, \ldots, x_n^{(k+1)})$
10. se convergência = .FALSE. (algoritmo 10.3) então
    - 10.1. se $k \geq n_{max}$ então parar, o processo não está a convergir
    - 10.2. se quer recomeçar e $k =$ múltiplo de n então ir para o passo 5
    - 10.3. senão calcular o escalar β (H.S.) :
      - 10.3.1. calcular o vector y: para $i = 1, \ldots, n$ calcular $y_i = g_i^{(k+1)} - g_i^{(k)}$
      - 10.3.2. calcular $den = \sum_{i=1}^{n} d_i^{(k)} y_i$, $num = \sum_{i=1}^{n} g_i^{(k+1)} y_i$ e $\beta = \frac{num}{den}$
      - 10.3.3. ir para o passo 6
11. terminar com $x^* \leftarrow (x_i^{(k+1)}, i = 1, \ldots, n)$ e $f(x^*) \leftarrow f^{(k+1)}$.

10.3.15 Implementação. Rotina GRACON

O comportamento da rotina GRACON, que implementa o algoritmo 10.10, relativo aos três problemas já apresentados anteriormente, em 10.3.11 e 10.3.13, é o que a seguir se descreve.

A precisão usada no algoritmo 10.3 do critério de paragem é de $\varepsilon = 10^{-5}$. Se for escolhido outro valor mais baixo, a precisão aumenta e como consequência mais iterações são necessárias até se verificar a paragem do processo iterativo.

Exemplo 10.18 O problema

$$min_{x\in\mathbf{R}^2}\ (x_1+1)^2+(x_2+1)^2$$

é resolvido ao fim de 2 iterações, atingindo-se o vector $(-1,-1)^T$, com $f^*=0$.

Exemplo 10.19 A solução $(0,0)^T$, com $f^*=0$ do problema

$$min_{x\in\mathbf{R}^2}\ 144x_1^2+4x_2^2-8x_1x_2$$

obtém-se ao fim de 14 iterações.

No entanto, para resolver

$$min_{x\in\mathbf{R}^2}\ x_1^2+x_2^2-\frac{1}{3}x_1x_2$$

o processo leva 24 iterações, tendo parado no ponto $(0.007707, 0.007707)^T$, em que f atinge o valor 0.000099.

10.4 Problemas

1. Dada a função $f:\mathbf{R}\to\mathbf{R}$ definida por

$$f(x)=\begin{cases} 2x^2-1-2x & \text{se } x\notin[-1,2] \\ 3 & \text{se } x\in[-1,2] \end{cases}$$

   calcule os pontos estacionários. Destes, quais são os minimizantes? Justifique.

2. Mostre que qualquer ponto da linha $x_2-2x_1=0$ é um minimizante de $f:\mathbf{R}^2\to\mathbf{R}$ definida por

$$f(x_1,x_2)=4x_1^2-4x_1x_2+x_2^2.$$

3. Verifique se o ponto $(0,-1)^T$ é minimizante da função

$$f(x_1,x_2)=x_1+x_2+\frac{1}{2}(x_1^2+x_2^2)-\frac{x_1+x_1^2+x_1^3}{1+x_1^4}.$$

   Justifique.

4. Determine os pontos estacionários de

$$f(x_1, x_2) = (x_1 + x_2 - 4)^2 + (x_2^2 - 6x_2 + 8)^2.$$

Quantos pontos estacionários são minimizantes de f? Justifique.

N.B. Os zeros do polinómio $2x^3 - 18x^2 + 52x - 48$ são $2, 3$ e 4.

5. Dada a função $f : \mathbf{R} \to \mathbf{R}$ definida por

$$f(x) = x^2 + 4|x - 1|$$

calcule o seu mínimo usando o algoritmo de Fibonacci. O processo iterativo deve ser iniciado com $N = 10$ e pelo intervalo $[0.7, 1.3]$.

Considere $\epsilon = 0.0000001$.

6. Dada a função $f : \mathbf{R} \to \mathbf{R}$ definida por

$$f(x) = \begin{cases} |x| & \text{para } x < 0 \\ x^2 - x & \text{para } x \geq 0 \end{cases}$$

calcule o seu mínimo usando o algoritmo de Fibonacci. O processo iterativo deve ser iniciado com o intervalo:

$$[-2, 2].$$

Considere $N = 8$ e $\epsilon = 0.0000001$.

7. Dada a função $f : \mathbf{R} \to \mathbf{R}$ definida por

$$f(x) = x^2 - x\,,$$

calcule o seu mínimo usando o algoritmo de Davies, Swann e Campey (DSC), baseado na interpolação quadrática. O processo deve ser iniciado com o ponto $x_1 = 2$.

Considere $\delta = 1$, $M = 0.5$ e $\varepsilon = 0.5$.

8. Considere a função $f : \mathbf{R} \to \mathbf{R}$ definida por

$$f(x) = \begin{cases} (x-1)^2 & \text{se } x \notin [0, 2] \\ 1 & \text{se } x \in [0, 2] \end{cases}$$

Implemente o algoritmo de D.S.C. baseado em interpolação quadrática, fazendo $x_1 = 4$, $\delta = 0.5$ e $M = 0.5$.

9. Dado o problema

$$min_{x \in \mathbf{R}^2}\, 4(x_1 - 5)^2 + (x_2 - 6)^2,$$

calcule a solução, usando o método de procura de Rosenbrock, tomando como ponto de partida do processo o vector $(8, 9)^T$. Para a paragem do processo iterativo use $R < 0.05$. Considere i) $\delta = (1, 1)^T$ ii) $\delta = (0.1, 0.1)^T$. O valor mínimo de f é 0 e atinge-o no ponto $(5, 6)^T$. Qual das duas implementações foi mais rápida? Comente.

10. Calcule o mínimo da função $f(x)$ definida por

$$f(x_1, x_2) = \text{máx}[(x_1 - 1)^2, x_1^2 + 4(x_2 - 1)^2]$$

implementando

a) O método de procura de Rosenbrock, considerando o vector inicial $x^{(1)} = (-1, 0)^T$ e $(\delta_1, \delta_2) = (1, 1)$. Verifique que o processo aceita, logo na primeira iteração, os pontos $(0, 0)^T$ e $(0, 1)^T$ e calcula, pela primeira vez, novas direcções de procura no fim da segunda iteração. Considerando a condição de paragem apresentada no algoritmo 10.4, verifique que o processo pára ao fim de 88 iterações, tendo atingido o ponto $(0.499185, 1.020184)^T$ com $f = 0.250816$.

b) O método de Nelder-Mead, tomando para conjunto inicial os vectores

$$x_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad x_2 = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad \text{e} \quad x_3 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

e $\varepsilon = 10^{-3}$.

11. Dada a função $f : \mathbf{R}^2 \to \mathbf{R}$ definida por

$$f(x_1, x_2) = x_1^2 + x_2^2 - 2x_1x_2,$$

calcule o seu mínimo usando o método de Nelder-Mead.

O processo iterativo deve terminar quando o critério de paragem for verificado para $\varepsilon = 0.1$. Considere os seguintes pontos iniciais:

$$(0.1, -0.1)^T \quad (-0.1, -0.1)^T \quad (0.1, 0.1)^T$$

12. Dada a função $f : \mathbf{R}^2 \to \mathbf{R}$ definida por

$$f(x_1, x_2) = -x_1^2 - 6x_2^2,$$

calcule o seu máximo usando o método de Nelder-Mead. O processo iterativo deve terminar quando o critério de paragem for verificado para $\varepsilon = 0.1$. Considere os seguintes pontos iniciais:

$$(0.1, 0.1)^T \quad (-0.1, 0)^T \quad (0, -0.1)^T$$

13. Dada a função $f(x_1, x_2, x_3) = x_1^2 - x_1x_2 + x_2^2 - x_2x_3 + x_3^2 + 2x_1 - x_2$ de $\mathbf{R}^3$, calcule o seu mínimo, usando o método de procura de Nelder-Mead. Além do ponto inicial $(0, -1, 1)^T$, considere os seguintes pontos: $(1, 1, 1)^T$, $(1, 1, 0)^T$ e $(0, -1, 0)^T$.

14. Dada a função $f : \mathbf{R}^2 \to \mathbf{R}$ definida por

$$f(x_1, x_2) = x_1 x_2^2 + (2 - x_1)^2,$$

calcule o seu mínimo usando o método de segurança de Newton.

O processo iterativo deve ser iniciado com o ponto $(1, 1)^T$ e deve terminar quando o critério de paragem de Himmelblau for verificado para $\varepsilon_1 = \varepsilon_2 = \varepsilon_3 = 0.01$.

Tome $\eta = 0.00000001$.

Deve, também, implementar o algoritmo das repetidas divisões de α por dois, para calcular o comprimento do passo, em cada iteração.

15. Dada a função $f : \mathbf{R}^2 \to \mathbf{R}$ definida por

$$f(x_1, x_2) = (x_1 - 1)^2 + (x_2 - 1)^3,$$

calcule o seu mínimo usando o algoritmo de segurança de Newton.

O processo iterativo deve ser iniciado com o ponto $(2, 1)^T$ e deve terminar quando o critério de paragem de Himmelblau for verificado para $\varepsilon_1 = \varepsilon_2 = \varepsilon_3 = 0.01$.

Deve, também, implementar o algoritmo D.S.C. baseado na interpolação quadrática, com $\delta = 1$, $\alpha_1 = 0$ e $\varepsilon = 1$, para calcular o valor de α (comprimento do deslocamento), em cada iteração.

16. Considere o problema:

$$min_{x \in \mathbf{R}^2}\, 4 + \frac{9}{2}x_1 - 4x_2 + x_1^2 + 2x_2^2 - 2x_1 x_2 + x_1^4 - 2x_1^2 x_2.$$

Um dos minimizantes é $\approx (-1.05274, 1.02776)^T$ e o valor de $f^* \approx -0.51341$.

Compare o comportamento dos seguintes métodos de primeira ordem baseados no cálculo do gradiente: i) método de descida máxima ii) método Quasi-Newton iii) método dos gradientes conjugados, considerando as três aproximações iniciais:

$$x^{(1)} = \begin{pmatrix} -1 \\ 0 \end{pmatrix}, \quad \bar{x}^{(1)} = \begin{pmatrix} 0 \\ 3 \end{pmatrix} \quad \text{e} \quad \hat{x}^{(1)} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

Tome para ε (do critério de paragem no algoritmo 10.3) o valor 0.01.

17. Resolva o problema anterior usando o método de segurança de Newton. Tome os mesmos vectores iniciais e igual valor para ε. Confirme os seguintes resultados: 3 iterações se o processo for iniciado por $x^{(1)}$; 6 iterações se o processo for iniciado por $\bar{x}^{(1)}$ e 4 iterações para convergir para o outro mínimo $(1.941013, 3.854272)^T$, com $f = 0.985554$ quando o vector de partida do processo for $\hat{x}^{(1)}$.

18. Dada a função $f : \mathbf{R}^2 \to \mathbf{R}$ definida por

$$f(x_1, x_2) = x_2^2 - x_1^3 - 3x_1^2$$

determine o ponto solução do problema

$$min_{x \in \mathbf{R}^2} f(x_1, x_2),$$

usando para aproximação inicial o ponto $(1, 1)^T$ e implementando o algoritmo de segurança de Newton.

19. Implemente o método dos gradientes conjugados, na versão mais adequada, na resolução do problema

$$min_{x \in \mathbf{R}^2} 8x_1^2 - 4x_1x_2 + 5x_2^2.$$

Considere como aproximação inicial o ponto $(1, 1)^T$. Explique porque razão obteve a solução tão depressa.

20. Dada a função $f : \mathbf{R}^2 \to \mathbf{R}$ definida por

$$f(x_1, x_2) = x_1x_2^2 + (2 - x_1)^2,$$

calcule o seu mínimo usando o método dos gradientes conjugados.

O processo iterativo deve ser iniciado com o ponto $(1, 1)^T$ e deve terminar quando o critério de paragem de Himmelblau for verificado para $\varepsilon_1 = \varepsilon_2 = \varepsilon_3 = 0.5$.

Deve, também, implementar o algoritmo das repetidas divisões de α por dois para calcular o valor de α (comprimento do deslocamento), em cada iteração.

21. Dada a função $f : \mathbf{R}^2 \to \mathbf{R}$ definida por

$$f(x_1, x_2) = x_1^2 + x_2^2 - 2x_1x_2,$$

calcule o seu mínimo usando a versão mais adequada do método dos gradientes conjugados.

O processo iterativo deve ser iniciado com o ponto $(1, -1)^T$ e deve terminar quando o critério de paragem for verificado para $\varepsilon_1 = \varepsilon_2 = \varepsilon_3 = \varepsilon = 0.01$.

Deve, também, implementar o algoritmo das repetidas divisões de α por dois, para calcular o comprimento do passo, em cada iteração.

22. Considere o seguinte problema

$$min_{x \in \mathbf{R}^2} 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

baseado na função de Rosenbrock.

a) Compare o comportamento dos métodos de segurança de Newton, Quasi-Newton e gradientes conjugados na sua resolução, tomando o valor 10^{-3} para a tolerância ε do critério de paragem (algoritmo 10.3).

Considere $\mu_1 = 0.001$ e $\mu_2 = 0.9$. Tome como aproximação inicial o vector $(-1.2, 1.0)^T$.

b) Repita os testes de a), apenas para os métodos de segurança de Newton e gradientes conjugados, tomando agora $\varepsilon = 10^{-6}$.

23. Dada a função de Powell $f : \mathbf{R}^4 \to \mathbf{R}$ definida por

$$f(x_1, x_2, x_3, x_4) = (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4,$$

determine o ponto solução do problema

$$min_{x \in \mathbf{R}^4} f(x_1, x_2, x_3, x_4),$$

usando:

a) O método de segurança de Newton e considerando o vector inicial $(3, -1, 0, 1)^T$. Para $\mu_1 = 0.001$, $\mu_2 = 0.9$ e $\varepsilon = 10^{-3}$ verifique que o processo converge ao fim de 18 iterações, para $(0.001611, -0.000161, 0.000258, 0.000258)^T$ com $f = 0.000000$.

b) O método Quasi-Newton, com o mesmo vector inicial.

c) O método dos gradientes conjugados, com o mesmo vector inicial.

d) O método das direcções de descida máxima e com o mesmo vector inicial. Repita este método, usando agora $\mu_1 = 0.5$ e $\mu_2 = 0.9$.

Capítulo 11

Equações diferenciais com derivadas parciais

11.1 Introdução

11.1.1 Forma geral do problema

Neste Capítulo considera-se o cálculo numérico de uma aproximação à solução $u = u(x,t)$, de um caso especial, muito comum, de equação diferencial com derivadas parciais de segunda ordem

$$A(x,t)\frac{\partial^2 u}{\partial t^2}+B(x,t)\frac{\partial^2 u}{\partial t \partial x}+C(x,t)\frac{\partial^2 u}{\partial x^2}+D(x,t)\frac{\partial u}{\partial t}+E(x,t)\frac{\partial u}{\partial x}+F(x,t)u = G(x,t). \quad (11.1)$$

Os coeficientes $A, B, \ldots$ são funções contínuas numa região Ω do plano XOT e a função $u(x,t)$ e/ou as suas derivadas devem verificar um conjunto de condições na fronteira de Ω.

Teoricamente é possível mostrar a existência e unicidade da solução, no entanto, estes resultados dependem das características do problema, tais como, o tipo da equação, as condições na fronteira e a geometria da região, Ω.

Faremos uma descrição sumária de alguns dos métodos mais simples para resolver equações com derivadas parciais. Embora os problemas mais complexos necessitem de métodos mais complicados e específicos, os aqui descritos são adequados para um número substancial de aplicações.

11.1.2 Características do problema. Classificação das equações

Algumas das características das equações diferenciais com derivadas parciais relacionam-se com a *ordem*, a *dimensão* e a *linearidade*.

A *ordem* está relacionada com a ordem mais elevada da derivada da função $u(x,t)$ envolvida na equação. A formulação (11.1) define uma equação de segunda ordem.

A *dimensão* refere-se ao número de variáveis *espaço* presentes na equação. O exemplo da equação (11.1) tem dimensão *um* e a variável espaço é o x. A outra variável, t, refere-se ao *tempo*, e é outra das variáveis independentes.

Uma equação diz-se *linear* se a função solução, u, e as suas derivadas aparecem envolvidas em termos lineares na equação diferencial.

Há maneiras diferentes de classificar as *equações diferenciais com derivadas parciais*. Uma delas considera o sinal da expressão

$$\Delta = B^2(x,t) - 4A(x,t)C(x,t).$$

i) Se $\Delta = 0$ no domínio Ω, a equação diz-se *parabólica*.

Um exemplo típico é

$$\frac{\partial u}{\partial t} = C\frac{\partial^2 u}{\partial x^2},$$

com C constante e positivo, conhecido por equação do calor ou equação de difusão e descreve a distribuição da temperatura, $u(x,t)$, numa barra ou a difusão de gases, ao longo do tempo, t.

ii) Se $\Delta > 0$ em Ω, a equação diz-se *hiperbólica*.

Um exemplo típico é a equação das ondas

$$\frac{\partial^2 u}{\partial t^2} = C\frac{\partial^2 u}{\partial x^2}$$

com C constante e positivo e descreve a posição vibratória de uma mola ou a vibração longitudinal numa barra, ao longo do tempo.

iii) Se $\Delta < 0$ em Ω, a equação é *elíptica* e dois exemplos são:

- a equação de Laplace
$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$$
- e, a equação de Poisson
$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = G(x,y).$$

Estas equações são usadas para descrever a distribuição, independente do tempo ($\frac{\partial u}{\partial t} = 0$), da temperatura numa placa. A solução analítica de uma equação elíptica a duas dimensões é uma função, $u(x,y)$, das coordenadas do espaço, x e y.

Os casos i) e ii) são dependentes do tempo e são conhecidos por *problemas com condições iniciais*. A solução é conhecida no instante $t = 0$ (condição inicial) e pretende-se saber como é que ela se propaga no espaço ao longo do tempo.

Outra característica deste tipo de problemas refere-se ao facto do domínio ser aberto ao longo do tempo.

A outra categoria de equações com derivadas parciais, descrita em iii), define os *problemas com condições de fronteira* em que o domínio está cercado por condições nos extremos ou na sua fronteira.

Também existem vários tipos de condições de fronteira:

i) se a condição especifica o valor de $u(x, y)$ na fronteira do domínio, então ela diz-se de *primeira espécie* ou *condição de Dirichlet*;

ii) se a condição especifica a derivada normal, $\frac{\partial u}{\partial n}$, na fronteira, então a condição diz-se de *segunda espécie* ou de *Neumann*;

iii) se a condição de fronteira especifica $\gamma u + \frac{\partial u}{\partial n}$, ela é de *terceira espécie* e tem solução única se $\gamma > 0$.

Estas características reflectem-se nos métodos numéricos. Para os problemas com condições iniciais, a solução numérica constrói-se passo a passo, ao longo do tempo, enquanto que nos problemas com condições de fronteira a solução é construída simultaneamente em todo o domínio.

11.1.3 Tipos de métodos

A primeira etapa na resolução de uma equação diferencial com derivadas parciais é a sua *discretização*. Um dos métodos mais simples de discretização consiste em substituir as derivadas, envolvidas na equação diferencial e nas condições de fronteira e iniciais, por *diferenças finitas*.

Estas diferenças finitas podem ser

- *descendentes* definidas por $\Delta u_j = u_{j+1} - u_j$ (relembrar o que foi dito em 6.2.2 do Capítulo 6),
- *ascendentes*, $\nabla u_j = u_j - u_{j-1}$ (em 6.2.3), ou
- *centrais*, de acordo com a seguinte definição: $\delta u_j = u_{j+\frac{1}{2}} - u_{j-\frac{1}{2}}$ (veja-se em 6.2.4).

O domínio é também substituído por um conjunto finito de pontos que definem uma malha e o valor da função u é calculado nos pontos da malha, com está indicado na figura 11.1.

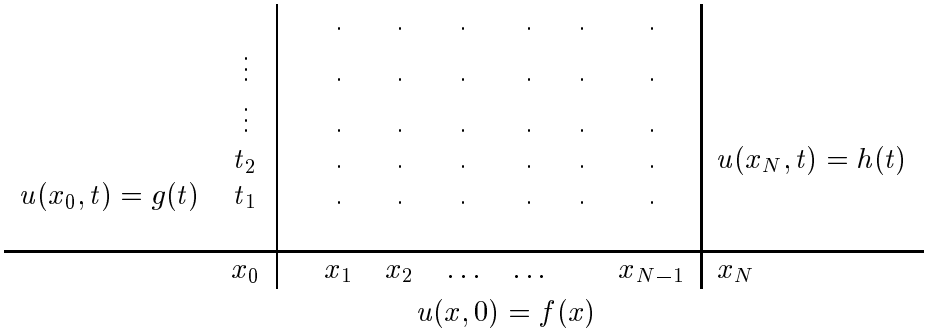


Figura 11.1: Malha de pontos no domínio

11.2 Equações parabólicas

11.2.1 Introdução teórica

Considere a equação

$$\frac{\partial u}{\partial t} = C\frac{\partial^2 u}{\partial x^2}, \;\; 0 \leq x \leq l, \;\; \text{e} \;\; t \geq 0, \tag{11.2}$$

com C constante, em que as condições iniciais e os valores nos extremos são

$$\begin{aligned} u(x,0) &= f(x), \\ u(0,t) &= g(t), \\ u(l,t) &= h(t). \end{aligned} \tag{11.3}$$

De acordo com a malha representada na figura 11.1, suponha que o intervalo entre os pontos, ao longo da direcção x, é constante e igual a Δx, com $\Delta x = \frac{l}{N}$ (N inteiro) e, ao longo da direcção t, o passo é também constante e igual a Δt.

Ao longo da direcção x existem $N+1$ pontos. Os *pontos da malha* são designados por

$$(x_i, t_n) \;\; \text{com} \;\; x_i = i\Delta x, \;\; t_n = n\Delta t, \;\; i = 0, \ldots, N, \;\; n = 0, 1, \ldots \tag{11.4}$$

e os *valores da função* $u(x,t)$, ao longo da malha, são aproximados por

$$u_i^n \approx u(x_i, t_n). \tag{11.5}$$

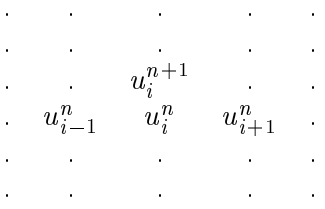


Figura 11.2: Pontos do esquema explícito para equações parabólicas

11.2.2 Esquema explícito

Um dos métodos numéricos de discretização mais simples substitui as derivadas da equação parabólica (11.2) por aproximações baseadas em diferenças finitas, pressupondo que são conhecidos os valores de u no instante $t = t_n$, mas não são conhecidos os do instante $t = t_{n+1}$. Assim

$$\frac{\partial u}{\partial t} \approx \frac{u_i^{n+1} - u_i^n}{\Delta t},$$

o que equivale a ter usado a diferença descendente de primeira ordem, em relação à variável t, e

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{(\Delta x)^2},$$

correspondendo a uma discretização baseada na diferença central de segunda ordem, em relação a x.

A equação discretizada reduz-se a

$$u_i^{n+1} = u_i^n + \frac{C\Delta t}{(\Delta x)^2}(u_{i+1}^n - 2u_i^n + u_{i-1}^n), \quad \text{para } i = 1, \ldots, N-1 \text{ e } n = 0, \ldots \tag{11.6}$$

e as condições a

$$u_i^0 = f(x_i), \;\; u_0^n = g(t_n) \;\text{ e }\; u_N^n = h(t_n). \tag{11.7}$$

Uma fórmula, como esta, que explicita um valor de u desconhecido, no instante $t = t_{n+1}$, em função de vários valores conhecidos de u, no instante $t = t_n$, diz-se *explícita*.

A figura 11.2. ilustra os pontos envolvidos neste esquema explícito, onde se calcula, para $x_i = i\Delta x$, uma aproximação de cada vez, à função u, no instante $t = t_{n+1}$.

11.2.3 Erro de truncatura do esquema explícito

Uma outra maneira, de representar a equação diferença (11.6), utiliza a notação de operador, baseado no operador diferença $\mathbf{L}$

$$\mathbf{L}u_i^n = (\Delta_t - \frac{C\Delta t}{(\Delta x)^2}\delta_x^2)u_i^n = 0, \tag{11.8}$$

em que Δ_t é a diferença descendente de primeira ordem, relativa a t, e δ_x^2 é a diferença central de segunda ordem, relativa a x.

Define-se *erro de truncatura* do método como sendo o resultado da aplicação do operador diferença à solução exacta $u(x,t)$,

$$e_T(x,t) = \mathbf{L}u(x,t). \tag{11.9}$$

De acordo com a expansão em série de Taylor, tem-se

$$u(x,t+\Delta t) = u(x,t) + \Delta t \frac{\partial u}{\partial t} + \frac{1}{2}(\Delta t)^2 \frac{\partial^2 u}{\partial t^2} + \cdots,$$

$$u(x \pm \Delta x, t) = u(x,t) \pm \Delta x \frac{\partial u}{\partial x} + \frac{1}{2}(\Delta x)^2 \frac{\partial^2 u}{\partial x^2} \pm \cdots$$

ou seja

$$\frac{u(x,t+\Delta t) - u(x,t)}{\Delta t} = \frac{\partial u}{\partial t} + \frac{1}{2}\Delta t \frac{\partial^2 u}{\partial t^2} + \cdots$$

e

$$\frac{u(x+\Delta x,t) - 2u(x,t) + u(x-\Delta x,t)}{(\Delta x)^2} = \frac{\partial^2 u}{\partial x^2} + \frac{1}{12}(\Delta x)^2 \frac{\partial^4 u}{\partial x^4} + \cdots$$

Se substituirmos estas duas expressões em (11.9) e usarmos a equação parabólica (11.2), obtém-se para o erro de truncatura

$$e_T(x,t) = \frac{1}{2}\Delta t \frac{\partial^2 u}{\partial t^2} - \frac{1}{12} C (\Delta x)^2 \frac{\partial^4 u}{\partial x^4} + \cdots$$

ou, ainda, se usarmos novamente (11.2),

$$e_T(x,t) = \frac{1}{2} C^2 \Delta t \, (1 - \frac{1}{6r}) \frac{\partial^4 u}{\partial x^4} + \cdots \tag{11.10}$$

sendo $r = \frac{C\Delta t}{(\Delta x)^2}$.

Um limite superior para o erro é então

$$|e_T(x,t)| \leq K \Delta t \tag{11.11}$$

se $K = \frac{1}{2}C^2(1 - \frac{1}{6r})M > 0$ e M for um majorante da quarta derivada de u em ordem a x.

A partir da equação (11.10), tem-se

$$e_T(x,t) = \mathcal{O}(\Delta t) \text{ quando } \Delta t \to 0,$$

o que implica que este esquema é de *primeira ordem*. Note-se que, no caso particular de $r = \frac{1}{6}$, o termo dominante em (11.10) anula-se e o esquema passa a ser de segunda ordem.

11.2.4 Estudo da convergência do esquema explícito

Seja $u(x,t)$ a solução exacta da equação diferencial com derivadas parciais, com variáveis independentes x e t e, seja, ainda, u_i^n a solução exacta da equação diferença (método numérico) usada para aproximar a equação diferencial. A solução, obtida pela equação diferença, diz-se *convergente*, se u_i^n tende para $u(x_i,t_n)$ quando Δt e Δx tendem para zero.

O estudo da convergência torna-se, na maior parte dos casos, difícil pois a expressão para o erro de discretização vem em função de derivadas, para as quais não se conhecem limites.

À diferença entre $u(x_i,t_n)$ e o valor exacto da equação (11.6), u_i^n,

$$\epsilon_i^n = u(x_i,t_n) - u_i^n,$$

chama-se *erro de discretização* e satisfaz

$$\epsilon_i^{n+1} = \epsilon_i^n + r\,\delta_x^2\,\epsilon_i^n + \Delta t\; e_T(x_i,t_n) = r\epsilon_{i+1}^n + (1-2r)\epsilon_i^n + r\epsilon_{i-1}^n + \Delta t\; e_T(x_i,t_n)$$

se usarmos (11.6) e (11.9).

Se o erro máximo, no instante $t = t_n$ é dado por $E^n = mx_i|\epsilon_i^n|$, então

$$E^{n+1} \leq mx_i|r\,\epsilon_{i+1}^n + (1-2r)\epsilon_i^n + r\epsilon_{i-1}^n| + \Delta t\; mx_i|e_T(x_i,t_n)|.$$

Como, para $0 \leq r \leq \frac{1}{2}$ todos os coeficientes dos $\epsilon' s$ são não negativos ($C > 0$), tem-se

$$E^{n+1} \leq E^n + K(\Delta t)^2$$

usando o limite superior do erro de truncatura apresentado em (11.11).

Como os valores iniciais $u(x_i,t_0)$ e u_i^0 são iguais, $E^0 = 0$, e

$$E^n \leq nK(\Delta t)^2 = t_n K\Delta t \to 0 \text{ quando } \Delta t \to 0, \tag{11.12}$$

o que prova a *convergência da solução, quando* $0 \leq r \leq \frac{1}{2}$.

11.2.5 Estabilidade do esquema explícito

Quando se usa um esquema/método numérico na resolução de um problema de equações diferenciais com derivadas parciais, é a equação diferença, que define o esquema, que é efectivamente resolvida.

Como os cálculos são realizados em aritmética discreta (cada número é representado por uma sequência finita de dígitos), de cada vez que a equação é usada introduzem-se erros de arredondamento. A solução calculada não é exactamente u_i^n, mas é antes U_i^n, conhecida por *solução numérica*.

Uma equação diferença (método numérico) diz-se *estável* (numericamente estável) se a acumulação dos erros de arredondamento cometidos ao longo dos passos se considera

insignificante. Assim, para analisar a estabilidade de uma equação diferença temos de analisar a acumulação dos erros cometidos durante os cálculos.

Um dos processos, normalmente usados para esta análise, reduz as equações diferença à forma matricial e examina os valores próprios de uma matriz associada ao processo.

Para o esquema explícito da equação (11.6), se fizermos $i = 1, \ldots, N-1$, obtém-se

$$\begin{aligned} u_1^{n+1} &= (1-2r)u_1^n + ru_2^n + rg(t_n) \\ u_2^{n+1} &= ru_1^n + (1-2r)u_2^n + ru_3^n \\ \ldots &= \ldots \\ u_{N-1}^{n+1} &= ru_{N-2}^n + (1-2r)u_{N-1}^n + rh(t_n) \end{aligned}$$

que pode ser apresentado como

$$\begin{pmatrix} u_1^{n+1} \\ u_2^{n+1} \\ u_3^{n+1} \\ \vdots \\ u_{N-1}^{n+1} \end{pmatrix} = \begin{pmatrix} 1-2r & r & 0 & 0 & \ldots & 0 & 0 \\ r & 1-2r & r & 0 & \ldots & 0 & 0 \\ 0 & r & 1-2r & r & \ldots & 0 & 0 \\ & & & & \ddots & & \\ 0 & 0 & 0 & 0 & \ldots & r & 1-2r \end{pmatrix} \begin{pmatrix} u_1^n \\ u_2^n \\ u_3^n \\ \vdots \\ u_{N-1}^n \end{pmatrix} + r \begin{pmatrix} g(t_n) \\ 0 \\ 0 \\ \vdots \\ h(t_n) \end{pmatrix}$$

ou

$$u_\cdot^{n+1} = Au_\cdot^n + rg_n$$

em que $u_\cdot^{n+1}$, $u_\cdot^n$ e $g_n = (g(t_n),\, 0,\, \ldots,\, 0,\, h(t_n))^T$ são vectores de $N-1$ elementos e A é a matriz quadrada e simétrica, de ordem $N-1$.

Se $u_\cdot^0$ for o vector dos valores iniciais, então

$$u_\cdot^{n+1} = Au_\cdot^n + rg_n = A(Au_\cdot^{n-1} + rg_{n-1}) + rg_n = A^2(Au_\cdot^{n-2} + rg_{n-2}) + Arg_{n-1} + rg_n$$

ou, continuando a substituir

$$u_\cdot^{n+1} = A^{n+1}u_\cdot^0 + A^n rg_0 + A^{n-1}rg_1 + \cdots + rg_n. \tag{11.13}$$

Suponha, agora, que se introduzem erros em cada ponto da malha, no instante $t = t_0$, e em vez de termos $u_\cdot^0$ passamos a ter $U_\cdot^0$. Os valores calculados passam a ser

$$U_\cdot^1 = AU_\cdot^0 + rg_0,$$

$$U_\cdot^2 = AU_\cdot^1 + rg_1 = A(AU_\cdot^0 + rg_0) + rg_1$$

$$\cdots$$

$$U_\cdot^{n+1} = A^{n+1}U_\cdot^0 + A^n rg_0 + A^{n-1}rg_1 + \cdots + rg_n. \tag{11.14}$$

Se definirmos o *vector erro de arredondamento*, no instante $t = t_{n+1}$, como

$$e_\cdot^{n+1} = u_\cdot^{n+1} - U_\cdot^{n+1}$$

então, de (11.13) e (11.14), chega-se a

$$e_{\cdot}^{n+1} = A^{n+1}(u_{\cdot}^{0} - U_{\cdot}^{0}) = A^{n+1}e_{\cdot}^{0}.$$

Para que o esquema (11.6) seja estável, os erros $e_{\cdot}^{n+1}$ devem permanecer limitados à medida que n aumenta indefinidamente.

Iremos analisar o comportamento destes erros, recorrendo à expressão do vector erro em função dos valores próprios de A, λ_j, $j = 1, \ldots, N-1$. Suponha que os $N-1$ valores próprios de A são distintos. Os vectores próprios associados, v_j, formam um conjunto de vectores linearmente independentes, em que $Av_j = \lambda_j v_j$, $j = 1, \ldots, N-1$.

O vector erro $e_{\cdot}^{0}$ pode exprimir-se como uma função dos vectores próprios,

$$e_{\cdot}^{0} = \sum_{j=1}^{N-1} \alpha_j v_j$$

sendo os α_j escalares a determinar. Assim,

$$e_{\cdot}^{1} = Ae_{\cdot}^{0} = A\sum_{j=1}^{N-1} \alpha_j v_j = \sum_{j=1}^{N-1} \alpha_j Av_j = \sum_{j=1}^{N-1} \alpha_j \lambda_j v_j$$

e, usando o mesmo raciocínio recursivo, conclui-se que

$$e_{\cdot}^{n+1} = \sum_{j=1}^{N-1} \alpha_j (\lambda_j)^{n+1} v_j,$$

o que prova que o erro não aumenta com n, se os valores próprios λ_j, de A, *forem, em valor absoluto, menores ou iguais a um.*

Para conhecermos os valores próprios da matriz A, basta colocá-la na forma:

$$A = \begin{pmatrix} 1 & & & & \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & \\ & & & & 1 \end{pmatrix} + r \begin{pmatrix} -2 & 1 & & & & & \\ 1 & -2 & 1 & & & & \\ & & & \ddots & & & \\ & & & & 1 & -2 & 1 \\ & & & & & 1 & -2 \end{pmatrix},$$

ou

$$A = I + rT$$

em que T é uma matriz tridiagonal, de ordem $N-1$, cujos valores e vectores próprios associados são conhecidos e respectivamente iguais a

$$-4sen^2(\frac{j\pi}{2N}), \tag{11.15}$$

e

$$(sen(\frac{j\pi}{N}),\ sen(\frac{2j\pi}{N}),\ \ldots,\ sen(\frac{(N-1)j\pi}{N}))^T$$

$$\text{para } j = 1, \ldots, N-1.$$

Assim, os valores próprios da matriz A são

$$\lambda_j = 1 - 4r\, sen^2(\frac{j\pi}{2N}),$$

e a *condição de estabilidade* do esquema (11.6) reduz-se a

$$|1 - 4r\, sen^2(\frac{j\pi}{2N})| \leq 1$$

ou

$$-1 \leq 1 - 4r\, sen^2(\frac{j\pi}{2N})$$

e

$$1 - 4r sen^2(\frac{j\pi}{2N}) \leq 1.$$

A segunda inequação é sempre verdadeira para $r \geq 0$ e da primeira tira-se que

$$r \leq \frac{1}{2\, sen^2(\frac{j\pi}{2N})}. \tag{11.16}$$

Desta análise, conclui-se que este *esquema explícito é estável para* $0 \leq r = \frac{C\Delta t}{(\Delta x)^2} \leq \frac{1}{2}$, que é o menor valor que a expressão (11.16) pode atingir. Isto significa que os valores escolhidos para Δt e Δx devem verificar a condição, se quisermos obter resultados com alguma precisão.

Exemplo 11.1 Na resolução da equação parabólica

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}, \;\; 0 \leq x \leq 1, \;\; \text{e} \;\; t \geq 0,$$

com

$$u(x,0) = x(1-x),$$
$$u(0,t) = 0,$$
$$u(1,t) = 0$$

e para $\Delta x = 0.25$ e $\Delta t = 0.05$ tem-se $r = 0.8$. O esquema explícito (11.6) reduz-se, na forma matricial (veja-se 11.2.5) a

$$u_{\cdot}^{n+1} = \begin{pmatrix} -0.6 & 0.8 & \\ 0.8 & -0.6 & 0.8 \\ & 0.8 & -0.6 \end{pmatrix} u_{\cdot}^{n} + 0.8 \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

para $n \geq 0$. Como o vector inicial é $u_{\cdot}^{0} = (0.1875, 0.25, 0.1875)^T$, obtém-se

n	0	1	2
t	0.05	0.1	0.15
$u_{\cdot}^{n+1}$	0.0875	0.0675	-0.0005
	0.15	0.05	0.078
	0.0875	0.0675	-0.0005

Se o valor de Δt descer para 0.025, o valor de r é agora igual a 0.4 ($< \frac{1}{2}$), a equação matricial do esquema é

$$u_{\cdot}^{n+1} = \begin{pmatrix} 0.2 & 0.4 & \\ 0.4 & 0.2 & 0.4 \\ & 0.4 & 0.2 \end{pmatrix} u_{\cdot}^{n} + 0.4 \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

e a solução é agora estável:

n	1	3	5
t	0.05	0.1	0.15
$u_{\cdot}^{n+1}$	0.1075	0.0627	0.036716
	0.15	0.0884	0.051888
	0.1075	0.0627	0.036716

11.2.6 Um esquema implícito simples

O esquema que a seguir se descreve é uma alternativa ao anterior. Embora não seja computacionalmente tão simples como o explícito, é convergente e estável para todos os valores positivos de r, o que o torna, em termos computacionais mais rápido e preciso. Não é necessário usar valores tão pequenos de Δt e $(\Delta x)^2$, como se costuma fazer quando se implementa o esquema de 11.2.2.

Em vez de tomar vários valores de u, já conhecidos, do instante $t = t_n$, usa vários valores desconhecidos do instante seguinte, $t = t_{n+1}$. A equação diferencial é aproximada por

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = C \frac{u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}}{(\Delta x)^2}$$

dando origem ao seguinte esquema *implícito*,

$$-ru_{i-1}^{n+1} + (1+2r)u_i^{n+1} - ru_{i+1}^{n+1} = u_i^n \quad \text{para} \quad i = 1, \ldots, N-1 \quad \text{e} \quad n = 0, \ldots \qquad (11.17)$$

em que $r = \frac{C\Delta t}{(\Delta x)^2}$ e as condições iniciais e de fronteira são, mais uma vez, dadas por

$$u_i^0 = f(x_i), \; u_0^n = g(t_n) \; \text{ e } \; u_N^n = h(t_n).$$

A figura 11.3. ilustra os pontos deste esquema que calcula o conjunto das aproximações a u, para $x_i = i\Delta x$, $i = 1, \ldots, N-1$, no instante $t = t_{n+1}$, de uma vez só, através da *resolução de um sistema de equações lineares*. Um método como este, em que o cálculo de u necessita da solução de um sistema de equações, é designado por *método implícito*.

Escrevendo uma equação (11.17) para cada $i, i = 1, \ldots, N-1$ obtém-se um sistema de equações lineares tridiagonal,

$$\begin{pmatrix} 1+2r & -r & 0 & \ldots & 0 & 0 & 0 \\ -r & 1+2r & -r & \ldots & 0 & 0 & 0 \\ & & & \ddots & & & \\ 0 & 0 & 0 & \ldots & -r & 1+2r & -r \\ 0 & 0 & 0 & \ldots & 0 & -r & 1+2r \end{pmatrix} \begin{pmatrix} u_1^{n+1} \\ u_2^{n+1} \\ \vdots \\ u_{N-2}^{n+1} \\ u_{N-1}^{n+1} \end{pmatrix} = \begin{pmatrix} u_1^n \\ u_2^n \\ \vdots \\ u_{N-2}^n \\ u_{N-1}^n \end{pmatrix} + r \begin{pmatrix} g(t_{n+1}) \\ 0 \\ \vdots \\ 0 \\ h(t_{n+1}) \end{pmatrix}$$

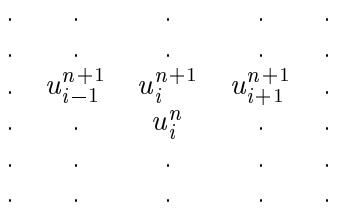


Figura 11.3: Pontos do esquema implícito para equações parabólicas

cuja solução nos dá todos os valores de u para aquele instante $t = t_{n+1}$. Quando $C > 0$ a matriz A dos coeficientes do sistema é estrita e diagonalmente dominante, é portanto não singular, e o sistema tem uma única solução. Como A é simétrica e definida positiva a factorização Cholesky (veja-se o algoritmo 3.7, em 3.5.3 do Capítulo 3) fornece um meio eficiente de resolução do sistema.

Se o número de equações do sistema for elevado, aconselha-se a utilização de um método iterativo (veja-se em 3.6.2 e 3.6.4).

O processo repete-se para os restantes instantes, separados por Δt.

11.2.7 A estabilidade do esquema implícito

Considere o sistema apresentado anteriormente, na forma

$$Au_{\cdot}^{n+1} = u_{\cdot}^{n} + rg_{n+1} \tag{11.18}$$

em que os vectores $u_{\cdot}^{n+1}$, $u_{\cdot}^{n}$ e $g_{n+1} = (g(t_{n+1}),\, 0,\, \ldots,\, 0,\, h(t_{n+1}))^T$ têm $N-1$ elementos e a matriz A dos coeficientes do sistema é quadrada, simétrica e de ordem $N-1$ e pode ser posta na forma

$$A = I - r \begin{pmatrix} -2 & 1 & & & & \\ 1 & -2 & 1 & & & \\ & & \ddots & & & \\ & & & 1 & -2 & 1 \\ & & & & 1 & -2 \end{pmatrix}$$

ou seja

$$A = I - rT. \tag{11.19}$$

T é a matriz tridiagonal, já evidenciada no caso explícito, em 11.2.5, cujos valores próprios são conhecidos.

Da equação (11.18) tira-se que

$$u_{\cdot}^{1} = A^{-1}u_{\cdot}^{0} + rA^{-1}g_1$$

$$u_{\cdot}^{2} = A^{-1}u_{\cdot}^{1} + rA^{-1}g_2 = (A^{-1})^2u_{\cdot}^{0} + r(A^{-1})^2g_1 + rA^{-1}g_2$$

$$\ldots$$

$$u_{\cdot}^{n+1} = (A^{-1})^{n+1}u_{\cdot}^{0} + r(A^{-1})^{n+1}g_1 + r(A^{-1})^{n}g_2 + \cdots + rA^{-1}g_{n+1}$$

e se $U_{\cdot}^{n+1}$ for a solução numérica, efectivamente calculada e afectada de erros de arredondamento, então a equação que relaciona o erro, de arredondamento, do passo $n+1$, $e_{\cdot}^{n+1} = u_{\cdot}^{n+1} - U_{\cdot}^{n+1}$, com o erro introduzido no vector inicial, $e_{\cdot}^{0}$, é a seguinte:

$$e_{\cdot}^{n+1} = (A^{-1})^{n+1}e_{\cdot}^{0}. \tag{11.20}$$

Usando um raciocínio idêntico ao usado no esquema explícito de 11.2.2. (veja-se em 11.2.5),

$$e_{\cdot}^{1} = A^{-1}\sum_{j=1}^{N-1}\alpha_j v_j = \sum_{j=1}^{N-1}\alpha_j A^{-1}v_j = \sum_{j=1}^{N-1}\alpha_j \lambda_j^{-1}v_j$$

em que λ_j, $j = 1, \ldots, N-1$, são os valores próprios de A e os vectores v_j são os vectores próprios de A^{-1} associados aos valores λ_j^{-1}, de tal forma que $A^{-1}v_j = \lambda_j^{-1}v_j$.

Nestas condições, de (11.20) conclui-se que

$$e_{\cdot}^{n+1} = \sum_{j=1}^{N-1}\alpha_j(\lambda_j^{-1})^{n+1}v_j$$

com $\lambda_j = 1 + 4r\,sen^2(\frac{j\pi}{2N})$, de acordo com (11.15), e a *condição de estabilidade* determina que

$$\left|\frac{1}{1 + 4r\ sen^2(\frac{j\pi}{2N})}\right| \leq 1.$$

Como esta inequação é sempre verdadeira para *qualquer valor positivo de* r $(C > 0)$, o *esquema implícito (11.17) é estável* sem haver necessidade de restringir demasiado os valores de Δt e Δx.

Exemplo 11.2 Considerando o problema do exemplo 11.1, tem-se para o esquema implícito (11.17), o seguinte sistema de equações matricial (veja-se em 11.2.6)

$$\begin{pmatrix} 2.6 & -0.8 & \\ -0.8 & 2.6 & -0.8 \\ & -0.8 & 2.6 \end{pmatrix} u_{\cdot}^{n+1} = u_{\cdot}^{n} + 0.8\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

para $n \geq 0$, considerando $\Delta x = 0.25$ e $\Delta t = 0.05$. Como o vector inicial é $u_{\cdot}^{0} = (0.1875, 0.25, 0.1875)^T$, obtém-se a solução

n	0	1	2
t	0.05	0.1	0.15
$u_{\cdot}^{n+1}$	0.125456	0.0848307	0.0576028
	0.173358	0.11888	0.0811708
	0.125456	0.0848307	0.0576028

$$\begin{array}{ccccc} . & . & . & . & . \\ . & . & . & . & . \\ . & u_{i-1}^{n+1} & u_i^{n+1} & u_{i+1}^{n+1} & . \\ . & u_{i-1}^{n} & u_i^{n} & u_{i+1}^{n} & . \\ . & . & . & . & . \\ . & . & . & . & . \end{array}$$

Figura 11.4: Pontos do esquema implícito de Crank-Nicholson

11.2.8 Esquema implícito de Crank-Nicholson

Os dois esquemas apresentados, o primeiro claramente explícito e o segundo implícito, são exemplos de duas situações extremas. A combinação de valores de u nos dois instantes $t = t_n$ e $t = t_{n+1}$ fornece o esquema implícito, muito conhecido, de *Crank-Nicholson*

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \frac{1}{2}\, C\{\frac{u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}}{(\Delta x)^2} + \frac{u_{i+1}^{n} - 2u_i^{n} + u_{i-1}^{n}}{(\Delta x)^2}\}$$

que origina a seguinte equação implícita,

$$-ru_{i-1}^{n+1} + (2+2r)u_i^{n+1} - ru_{i+1}^{n+1} = ru_{i-1}^{n} + (2-2r)u_i^{n} + ru_{i+1}^{n} \tag{11.21}$$

para $i = 1, \ldots, N-1$ e $n = 0, \ldots$, em que $r = \frac{C\Delta t}{(\Delta x)^2}$ e as condições iniciais e de fronteira são dadas por

$$u_i^0 = f(x_i),\ u_0^n = g(t_n)\ \text{ e }\ u_N^n = h(t_n).$$

As aproximações a u, para $x_i = i\Delta x$, $i = 1, \ldots, N-1$, no instante $t = t_{n+1}$, são calculadas a partir dos valores conhecidos no instante $t = t_n$, bem como, a partir dos desconhecidos do nível $n+1$. Torna-se assim necessário resolver um sistema para determinar simultaneamente os valores u_i^{n+1}, para $i = 1, \ldots, N-1$.

O esquema dos pontos necessários aos cálculos é o que está representado na figura 11.4.

O sistema das equações lineares resultante de (11.21) é o seguinte:

$$\begin{pmatrix} 2+2r & -r & 0 & \ldots & 0 & 0 & 0 \\ -r & 2+2r & -r & \ldots & 0 & 0 & 0 \\ & & & \ddots & & & \\ 0 & 0 & 0 & \ldots & -r & 2+2r & -r \\ 0 & 0 & 0 & \ldots & 0 & -r & 2+2r \end{pmatrix} \begin{pmatrix} u_1^{n+1} \\ u_2^{n+1} \\ \vdots \\ u_{N-2}^{n+1} \\ u_{N-1}^{n+1} \end{pmatrix} =$$

$$= \begin{pmatrix} 2-2r & r & 0 & \ldots & 0 & 0 & 0 \\ r & 2-2r & r & \ldots & 0 & 0 & 0 \\ & & & \ddots & & & \\ 0 & 0 & 0 & \ldots & r & 2-2r & r \\ 0 & 0 & 0 & \ldots & 0 & r & 2-2r \end{pmatrix} \begin{pmatrix} u_1^{n} \\ u_2^{n} \\ \vdots \\ u_{N-2}^{n} \\ u_{N-1}^{n} \end{pmatrix} + r \begin{pmatrix} g(t_{n+1}) + g(t_n) \\ 0 \\ \vdots \\ 0 \\ h(t_{n+1}) + h(t_n) \end{pmatrix}.$$

11.2.9 Estabilidade do método de Crank-Nicholson

A partir do sistema anterior, pode-se analisar o comportamento dos erros de arredondamento. Da forma

$$Au_{\cdot}^{n+1} = Bu_{\cdot}^{n} + rg_n^{n+1} \tag{11.22}$$

com o vector g_n^{n+1} definido por $(g(t_{n+1}) + g(t_n),\, 0,\, \ldots,\, 0,\, h(t_{n+1}) + h(t_n))^T$ e as matrizes A e B definidas por

$$A = 2I - rT \quad \text{e} \quad B = 2I + rT\,, \tag{11.23}$$

sendo T a matriz já anteriormente definida (veja-se em 11.2.5), tira-se que

$$\begin{aligned} u_{\cdot}^{n+1} &= A^{-1}Bu_{\cdot}^{n} + rA^{-1}g_n^{n+1} \\ &= A^{-1}B(A^{-1}Bu_{\cdot}^{n-1} + rA^{-1}g_{n-1}^{n}) + rA^{-1}g_n^{n+1} \\ &= \cdots \\ &= (A^{-1}B)^{n+1}u_{\cdot}^{0} + r(A^{-1})^{n+1}B^{n}g_0^1 + r(A^{-1})^{n}B^{n-1}g_1^2 + \cdots + r(A^{-1})^2Bg_{n-1}^{n} + rA^{-1}g_n^{n+1}. \end{aligned}$$

Se, nos cálculos, usarmos um vector inicial de valores perturbados, $U_{\cdot}^{0}$, os valores calculados subsequentes passam a ser

$$U_{\cdot}^{1} = A^{-1}BU_{\cdot}^{0} + rA^{-1}g_0^1$$

$$\begin{aligned} U_{\cdot}^{2} = A^{-1}BU_{\cdot}^{1} + rA^{-1}g_1^2 &= A^{-1}B[A^{-1}BU_{\cdot}^{0} + rA^{-1}g_0^1] + rA^{-1}g_1^2 \\ &= (A^{-1}B)^2U_{\cdot}^{0} + r(A^{-1})^2Bg_0^1 + rA^{-1}g_1^2 \end{aligned}$$

$$\ldots = \ldots$$

$$U_{\cdot}^{n+1} = (A^{-1}B)^{n+1}U_{\cdot}^{0} + r(A^{-1})^{n+1}B^{n}g_0^1 + \cdots + rA^{-1}g_n^{n+1}.$$

O *vector erro de arredondamento*, $e_{\cdot}^{n+1} = u_{\cdot}^{n+1} - U_{\cdot}^{n+1}$, reduz-se, neste caso, a

$$e_{\cdot}^{n+1} = (A^{-1}B)^{n+1}[u_{\cdot}^{0} - U_{\cdot}^{0}] = (A^{-1}B)^{n+1}e_{\cdot}^{0}, \tag{11.24}$$

ou, se substituirmos $e_{\cdot}^{0}$ pela sua expressão em função dos valores e vectores próprios de $A^{-1}B$, a

$$\begin{aligned} e_{\cdot}^{n+1} &= (A^{-1}B)^{n+1}\sum_{j=1}^{N-1}\alpha_j v_j \\ &= \sum_{j=1}^{N-1}\alpha_j(A^{-1}B)^{n+1}v_j \\ &= \sum_{j=1}^{N-1}\alpha_j\lambda_j^{n+1}v_j. \end{aligned}$$

Como os valores próprios de $A^{-1}B$, $\lambda_j, j = 1, \ldots, N-1$, vêm em função dos valores próprios da matriz T (em (11.15)), tem-se

$$\lambda_j = \frac{2 - 4rsen^2(\frac{j\pi}{2N})}{2 + 4rsen^2(\frac{j\pi}{2N})} \quad \text{para} \quad j = 1, 2, \ldots, N-1;$$

a *condição de estabilidade* determina que

$$|\frac{2 - 4rsen^2(\frac{j\pi}{2N})}{2 + 4rsen^2(\frac{j\pi}{2N})}| \leq 1, \tag{11.25}$$

o que equivale a

$$-1 \leq \frac{2 - 4rsen^2(\frac{j\pi}{2N})}{2 + 4rsen^2(\frac{j\pi}{2N})}$$

e

$$\frac{2 - 4rsen^2(\frac{j\pi}{2N})}{2 + 4rsen^2(\frac{j\pi}{2N})} \leq 1.$$

Como estas condições são sempre verdadeiras, para $r > 0\,(C > 0)$, conclui-se que este esquema de Crank-Nicholson é *estável para qualquer valor de r positivo.*

11.3 Equações hiperbólicas

11.3.1 Introdução teórica

A equação diferencial do tipo hiperbólica mais simples é de primeira ordem

$$\frac{\partial u}{\partial t} + E\frac{\partial u}{\partial x} = 0,$$

em que E é uma constante, $0 \leq x \leq l$ e $t \geq 0$.

Outra equação hiperbólica, conhecida por equação das ondas, de segunda ordem, tem a forma

$$\frac{\partial^2 u}{\partial t^2} = C\frac{\partial^2 u}{\partial x^2},$$

com C constante e positivo, para $0 \leq x \leq l$ e $t \geq 0$.

O método das características é a técnica que calcula as melhores soluções das equações hiperbólicas. Veja-se, por exemplo, em Nakamura (1991) e Smith (1975).

Quando a equação diferencial tem descontinuidades nos valores iniciais, a técnica mais aconselhável para a sua resolução é o método das características.

No entanto, quando o problema não tem descontinuidades, as equações hiperbólicas podem ser satisfatoriamente resolvidas por *métodos numéricos* baseados em *diferenças finitas* convergentes e estáveis. Focaremos apenas este tipo de métodos.

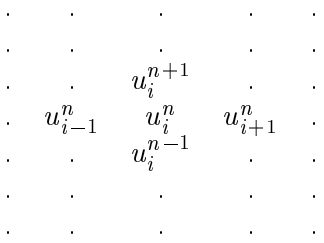


Figura 11.5: Os cinco pontos para as equações hiperbólicas

11.3.2 Esquema explícito

Para mostrar a utilização da técnica baseada na substituição das derivadas, envolvidas na equação, por diferenças finitas, comecemos com a seguinte equação hiperbólica de segunda ordem, o caso particular da equação das ondas:

$$\frac{\partial^2 u}{\partial t^2} = C\frac{\partial^2 u}{\partial x^2}, \;\; C > 0, \;\; 0 \leq x \leq l \;\; \text{e} \;\; t \geq 0, \tag{11.26}$$

em que as condições iniciais são

$$u(x,0) = f(x) \;\; \text{e} \;\; \frac{\partial u(x,0)}{\partial t} = g(x)$$

e as condições na fronteira

$$u(0,t) = u(l,t) = 0.$$

Considera-se, uma vez mais, $x_i = i\Delta x$, $i = 0, \ldots, N$ e $t_n = n\Delta t$, $n = 0, 1, \ldots$. Se aproximarmos as segundas derivadas, quer em ordem a t quer em ordem a x, por diferenças centrais explícitas, obtém-se

$$\frac{u_i^{n+1} - 2u_i^n + u_i^{n-1}}{(\Delta t)^2} = C\frac{(u_{i+1}^n - 2u_i^n + u_{i-1}^n)}{(\Delta x)^2}.$$

A equação diferença resultante, na sua forma final, reduz-se a

$$u_i^{n+1} = ru_{i-1}^n + 2(1-r)u_i^n + ru_{i+1}^n - u_i^{n-1} \tag{11.27}$$

para $i = 1, \ldots, N-1$ e $n = 0, \ldots$, em que $r = \frac{C(\Delta t)^2}{(\Delta x)^2}$ e as condições iniciais e de fronteira, também discretizadas, são

$$u_0^n = u_N^n = 0, \;\; u_i^0 = f(x_i) \;\; \text{e} \;\; \frac{u_i^1 - u_i^{-1}}{2\Delta t} = g(x_i). \tag{11.28}$$

A figura 11.5 apresenta os cinco pontos da malha envolvidos no cálculo de (11.27).

Da equação (11.27), e para $i = 1, \ldots, N-1$, resulta a seguinte forma matricial

$$u_{\cdot}^{n+1} = Au_{\cdot}^{n} - u_{\cdot}^{n-1},$$

o que é o mesmo que

$$\begin{pmatrix} u_1^{n+1} \\ u_2^{n+1} \\ u_3^{n+1} \\ \vdots \\ u_{N-1}^{n+1} \end{pmatrix} = \begin{pmatrix} 2(1-r) & r & 0 & 0 & \ldots & 0 & 0 \\ r & 2(1-r) & r & 0 & \ldots & 0 & 0 \\ 0 & r & 2(1-r) & r & \ldots & 0 & 0 \\ & & & & \ddots & & \\ 0 & 0 & 0 & 0 & \ldots & r & 2(1-r) \end{pmatrix} \begin{pmatrix} u_1^{n} \\ u_2^{n} \\ u_3^{n} \\ \vdots \\ u_{N-1}^{n} \end{pmatrix} - \begin{pmatrix} u_1^{n-1} \\ u_2^{n-1} \\ u_3^{n-1} \\ \vdots \\ u_{N-1}^{n-1} \end{pmatrix}$$

donde se tira que, para o cálculo do vector $u_{\cdot}^{n+1}$ precisamos de conhecer os vectores $u_{\cdot}^{n}$ e $u_{\cdot}^{n-1}$.

Da primeira vez que se usa este esquema, para $n = 0$, além de $u_{\cdot}^{0}$ precisamos de $u_{\cdot}^{-1}$ que pode ser calculado através de uma das condições de fronteira (11.28), resolvendo-a em ordem a $u_{\cdot}^{-1}$. Assim,

$$u_i^{-1} = u_i^1 - 2g(x_i)\Delta t$$

que substituído na equação (11.27), para $n = 0$, origina

$$u_i^1 = ru_{i-1}^0 + 2(1-r)u_i^0 + ru_{i+1}^0 - (u_i^1 - 2g(x_i)\Delta t).$$

A aproximação u_i^1, para cada i, $i = 1, \ldots, N-1$, é então calculada, de acordo com (11.28), através de

$$u_i^1 = \frac{r}{2}f(x_{i-1}) + (1-r)f(x_i) + \frac{r}{2}f(x_{i+1}) + g(x_i)\Delta t. \tag{11.29}$$

O estudo da convergência pode ser feito de um modo análogo ao apresentado em 11.2.4. O resultado do estudo concluiria que este esquema explícito, baseado nas diferenças centrais, é convergente para $r \leq 1$.

Relativamente à *estabilidade numérica* deste esquema explícito, e tal como se fez nos três métodos anteriores para as equações parabólicas (veja-se em (11.2.5), (11.2.7) e (11.2.9)), estabelecia-se a seguinte condição: *valores de r menores ou iguais a 1.*

Quando $r = 1$, $C(\Delta t)^2 = (\Delta x)^2$, os únicos erros que afectam os u_i^{n+1} calculados a partir de (11.27) são os erros de arredondamento.

11.3.3 Redução a duas equações de primeira ordem

A equação hiperbólica, apresentada em (11.26), pode ser resolvida numericamente reduzindo-a a duas equações de primeira ordem com posterior implementação dos métodos baseados em diferenças finitas.

Para isso, introduzem-se duas novas variáveis p e q definidas da seguinte maneira:

$$\frac{\partial u}{\partial x} = p \quad \text{e} \quad \frac{\partial u}{\partial t} = q.$$

A equação hiperbólica (11.26) e a relação

$$\frac{\partial^2 u}{\partial x \partial t} = \frac{\partial^2 u}{\partial t \partial x}$$

determinam o seguinte sistema de equações de primeira ordem

$$\frac{\partial q}{\partial t} = C\frac{\partial p}{\partial x} \quad \text{e} \quad \frac{\partial p}{\partial t} = \frac{\partial q}{\partial x}. \tag{11.30}$$

Qualquer fórmula de diferenças finitas que se use para aproximar as derivadas do sistema (11.30) deve definir um esquema convergente e estável.

O exemplo que aqui se apresenta utiliza *diferenças centrais* para as derivadas em ordem a x e *diferenças descendentes* para as derivadas em ordem a t. Posteriormente, q_i^n e p_i^n são substituídos pelas médias aritméticas de dois valores, um da sua direita e o outro da esquerda, em relação a x:

$$q_i^n \leftarrow \frac{1}{2}(q_{i+1}^n + q_{i-1}^n)$$

e

$$p_i^n \leftarrow \frac{1}{2}(p_{i+1}^n + p_{i-1}^n).$$

Tem-se, assim,

$$\frac{q_i^{n+1} - \frac{1}{2}(q_{i+1}^n + q_{i-1}^n)}{\Delta t} = \frac{C}{2\Delta x}(p_{i+1}^n - p_{i-1}^n)$$

e

$$\frac{p_i^{n+1} - \frac{1}{2}(p_{i+1}^n + p_{i-1}^n)}{\Delta t} = \frac{1}{2\Delta x}(q_{i+1}^n - q_{i-1}^n),$$

ou

$$q_i^{n+1} = C\,r\,(p_{i+1}^n - p_{i-1}^n) + \frac{1}{2}(q_{i+1}^n + q_{i-1}^n) \tag{11.31}$$

e

$$p_i^{n+1} = r(q_{i+1}^n - q_{i-1}^n) + \frac{1}{2}(p_{i+1}^n + p_{i-1}^n) \tag{11.32}$$

$$\text{para} \quad i = 1, \ldots, N-1,$$

e $r = \frac{\Delta t}{2\Delta x}$, com as condições $q_0^n = 0$, $p_0^n = 0$, $q_N^n = 0$, $p_N^n = 0$,

$$u_i^0 = f(x_i), \quad \frac{\partial u_i^0}{\partial x} = p_i^0 = \frac{\partial f(x_i)}{\partial x} \quad \text{e} \quad \frac{\partial u_i^0}{\partial t} = q_i^0 = g(x_i).$$

Na forma matricial, as equações (11.31) e (11.32), para $i = 1, \ldots, N-1$, são

$$q_\cdot^{n+1} = C\,A p_\cdot^n + B q_\cdot^n \tag{11.33}$$

e

$$p_\cdot^{n+1} = A q_\cdot^n + B p_\cdot^n \tag{11.34}$$

em que $p_{\cdot}^{n+1}$, $q_{\cdot}^{n+1}$, $p_{\cdot}^{n}$ e $q_{\cdot}^{n}$ são vectores de $N-1$ elementos e as matrizes A e B são definidas respectivamente por:

$$A = \begin{pmatrix} 0 & r & & & & & \\ -r & 0 & r & & & & \\ & -r & 0 & r & & & \\ & & & & \ddots & & \\ & & & & -r & 0 & r \\ & & & & & -r & 0 \end{pmatrix} \quad B = \begin{pmatrix} 0 & \frac{1}{2} & & & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & & & \\ & \frac{1}{2} & 0 & \frac{1}{2} & & & \\ & & & & \ddots & & \\ & & & & \frac{1}{2} & 0 & \frac{1}{2} \\ & & & & & \frac{1}{2} & 0 \end{pmatrix}.$$

Por substituição recursiva e usando as equações (11.33) e (11.34) tira-se que

$$\begin{aligned} q_{\cdot}^{n} &= C\,A(Aq_{\cdot}^{n-2} + Bp_{\cdot}^{n-2}) + B(C\,Ap_{\cdot}^{n-2} + Bq_{\cdot}^{n-2}) \\ &= (C\,A^2 + B^2)(C\,Ap_{\cdot}^{n-3} + Bq_{\cdot}^{n-3}) + 2C\,AB(Aq_{\cdot}^{n-3} + Bp_{\cdot}^{n-3}) \\ &= \ldots \end{aligned}$$

conseguindo-se para $p_{\cdot}^{n}$ uma relação idêntica.

Para a análise da *estabilidade*, começa-se por supôr que os valores iniciais de p e q, respectivamente $p_{\cdot}^{0}$ e $q_{\cdot}^{0}$, vêm afectados de erros e são efectivamente $P_{\cdot}^{0}$ e $Q_{\cdot}^{0}$. Tem-se, para os restantes valores calculados, a partir de (11.33) e (11.34),

$$Q_{\cdot}^{1} = C\,AP_{\cdot}^{0} + BQ_{\cdot}^{0} \quad \text{e} \quad P_{\cdot}^{1} = AQ_{\cdot}^{0} + BP_{\cdot}^{0},$$

$$Q_{\cdot}^{2} = 2C\,ABP_{\cdot}^{0} + (C\,A^2 + B^2)Q_{\cdot}^{0} \quad \text{e} \quad P_{\cdot}^{2} = 2ABQ_{\cdot}^{0} + (C\,A^2 + B^2)P_{\cdot}^{0},$$

$$Q_{\cdot}^{3} = (C^2A^3 + 3C\,AB^2)P_{\cdot}^{0} + (3C\,A^2B + B^3)Q_{\cdot}^{0}$$

e

$$P_{\cdot}^{3} = (C\,A^3 + 3AB^2)Q_{\cdot}^{0} + (3C\,A^2B + B^3)P_{\cdot}^{0},$$

$$\ldots$$

$$Q_{\cdot}^{n} = C[\sum_{j=0(impar)}^{n} \binom{n}{j} C^{\frac{j-1}{2}} A^j B^{n-j}]P_{\cdot}^{0} + [\sum_{j=0(par)}^{n} \binom{n}{j} C^{\frac{j}{2}} A^j B^{n-j}]Q_{\cdot}^{0}$$

e

$$P_{\cdot}^{n} = [\sum_{j=0(impar)}^{n} \binom{n}{j} C^{\frac{j-1}{2}} A^j B^{n-j}]Q_{\cdot}^{0} + [\sum_{j=0(par)}^{n} \binom{n}{j} C^{\frac{j}{2}} A^j B^{n-j}]P_{\cdot}^{0}\,.$$

Considerando os vectores dos erros de arredondamento, no instante $t = t_n$

$$e_{1\cdot}^{n} = q_{\cdot}^{n} - Q_{\cdot}^{n} \quad \text{e} \quad e_{2\cdot}^{n} = p_{\cdot}^{n} - P_{\cdot}^{n}$$

obtém-se a seguinte relação entre estes e os erros iniciais, na forma matricial,

$$\begin{pmatrix} e_{1\cdot}^{n} \\ e_{2\cdot}^{n} \end{pmatrix} = \begin{pmatrix} \sum_1 & C\sum_2 \\ \sum_2 & \sum_1 \end{pmatrix} \begin{pmatrix} e_{1\cdot}^{0} \\ e_{2\cdot}^{0} \end{pmatrix},$$

sendo

$$\sum_1 = \sum_{j=0(par)}^{n} \binom{n}{j} C^{\frac{j}{2}} A^j B^{n-j}$$

e

$$\sum_2 = \sum_{j=0(impar)}^{n} \binom{n}{j} C^{\frac{j-1}{2}} A^j B^{n-j}.$$

Usando o teorema de Gerschgorin, a *condição de estabilidade* determina que

$$\| \sum_1 + C \sum_2 \| \leq \| C(\sqrt{C}A + B)^n \| \leq 1$$

e

$$\| \sum_1 + \sum_2 \| \leq \| (\sqrt{C}A + B)^n \| \leq 1$$

uma vez que $\sum_1 + \sum_2 \leq (\sqrt{C}A + B)^n$ se $C \geq 1$. Como $\|B\|_\infty = 1$ então $(\|\sqrt{C}A\|)^2 \leq 1$ e $C(\|A\|)^2 \leq 1$. Daqui se tira que $2r \leq \frac{1}{\sqrt{C}}$ ou

$$\frac{\Delta t}{\Delta x} \leq \frac{1}{\sqrt{C}}.$$

11.4 Equações elípticas

11.4.1 Introdução teórica

O estudo das equações elípticas vai restringir-se ao seguinte problema com condições de fronteira

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + F(x,y)u(x,y) = G(x,y), \quad 0 \leq x \leq l \ \text{ e } \ 0 \leq y \leq k, \tag{11.35}$$

em que são especificadas as condições na fronteira do domínio

$$u(x,0) = f_1(x),$$

$$u(x,k) = f_2(x),$$

$$u(0,y) = g_1(y)$$

e

$$u(l,y) = g_2(y);$$

e as funções $F(x,y)$ e $G(x,y)$ são dadas, com $F(x,y) \leq 0$.

Quando $F(x,y) = G(x,y) = 0$ a equação (11.35) chama-se *equação de Laplace*; quando apenas $F(x,y) = 0$, (11.35) reduz-se à *equação de Poisson*; quando as funções $F(x,y)$ e $G(x,y)$ são diferentes de zero, tem-se então a *equação reduzida das ondas* ou *equação de Helmholtz*.

Para as equações elípticas consideramos apenas o problema de Dirichlet.

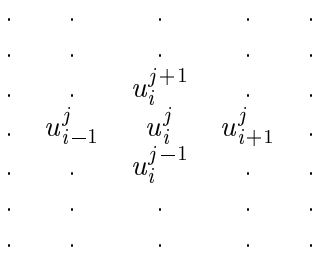


Figura 11.6: Os cinco pontos para as equações elípticas

11.4.2 Fórmula dos cinco pontos

Se fizermos $\Delta x = \frac{l}{N}$ e $\Delta y = \frac{k}{M}$, para N e M números inteiros, então os pontos da malha do domínio são definidos por

$$(x_i, y_j) \quad \text{com} \quad x_i = i\Delta x\,, i = 0, \ldots, N, \quad \text{e} \quad y_j = j\Delta y\,, j = 0, \ldots, M.$$

Vamos utilizar a seguinte notação simplificada:

$$u_i^j \approx u(x_i, y_j),$$

$$F_i^j \leftarrow F(x_i, y_j)$$

e

$$G_i^j \leftarrow G(x_i, y_j).$$

Se aproximarmos as segundas derivadas em ordem a x e a y da equação elíptica (11.35) por diferenças centrais explícitas, obtém-se

$$\frac{u_{i+1}^j - 2u_i^j + u_{i-1}^j}{(\Delta x)^2} + \frac{(u_i^{j+1} - 2u_i^j + u_i^{j-1})}{(\Delta y)^2} + F_i^j u_i^j = G_i^j.$$

Para simplificar, considera-se o espaçamento, entre pontos, constante e igual, ao longo de x e de y: $\Delta = \Delta x = \Delta y$.

A equação discretizada, na sua forma final, reduz-se ao esquema

$$u_i^{j+1} - (4 - (\Delta)^2 F_i^j)u_i^j + u_i^{j-1} + u_{i+1}^j + u_{i-1}^j = (\Delta)^2 G_i^j \tag{11.36}$$

para $i = 1, \ldots, N-1 \quad \text{e} \quad j = 1, \ldots, M-1$, conhecido por *fórmula dos cinco pontos*. As posições relativas dos pontos da fórmula estão representadas na figura 11.6.

Se $F(x, y) = G(x, y) = 0$, a equação diferença resultante da discretização da equação de Laplace,

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$$

é

$$u_i^{j+1} - 4u_i^j + u_i^{j-1} + u_{i+1}^j + u_{i-1}^j = 0 \tag{11.37}$$

e, para a equação de Poisson,

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = G(x,y)$$

o esquema baseado nos cinco pontos é

$$u_i^{j+1} - 4u_i^j + u_i^{j-1} + u_{i+1}^j + u_{i-1}^j = (\Delta)^2 G_i^j. \tag{11.38}$$

Uma vez que se usa uma equação, para cada ponto da malha no domínio, (11.36) ((11.37) ou (11.38)) origina um sistema de $(N-1)(M-1)$ equações lineares para a determinação do mesmo número de variáveis, u_i^j, $i = 1, \ldots, N-1$ e $j = 1, \ldots, M-1$.

Exemplo 11.3 Na resolução da equação elíptica

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = -2, \;\; 0 \leq x \leq 2, \;\; \text{e} \;\; 0 \leq y \leq 2,$$

com

$$u(x,0) = 0,$$
$$u(x,2) = 0,$$
$$u(0,y) = 0 \;\; \text{e} \;\; u(2,y) = 0$$

e para $\Delta x = \Delta y = 0.5$ tem-se a partir da fórmula dos cinco pontos (11.38) o seguinte sistema de $(N-1)(M-1) = 9$ equações lineares:

$$\begin{pmatrix} -4 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -4 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -4 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & -4 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & -4 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & -4 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & -4 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & -4 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -4 \end{pmatrix} u_{\cdot} = \begin{pmatrix} -0.5 \\ -0.5 \\ -0.5 \\ -0.5 \\ -0.5 \\ -0.5 \\ -0.5 \\ -0.5 \\ -0.5 \end{pmatrix}$$

com o vector $u_{\cdot}$ definido por

$$(u_1^1, u_2^1, u_3^1, u_1^2, u_2^2, u_3^2, u_1^3, u_2^3, u_3^3)^T.$$

A solução obtida pelo método iterativo de Gauss-Seidel (veja-se em 3.6.4), ao fim de 11 iterações (com $\varepsilon = 0.001$), é

$$(0.343369, 0.437119, 0.343559, 0.437119, 0.562119, 0.437309, 0.343559, 0.437309, 0.343655)^T.$$

11.4.3 Erro de truncatura e convergência

Para o esquema (11.36), baseado nos cinco pontos, é possível obter, usando um processo idêntico ao usado em 11.2.3 (para as equações parabólicas), a seguinte expressão para o erro de truncatura

$$e_T(x,y) = \frac{1}{12}((\Delta x)^2 \frac{\partial^4 u}{\partial x^4} + (\Delta y)^2 \frac{\partial^4 u}{\partial y^4}) + \cdots,$$

donde se conclui que é de *segunda ordem*.

A análise para estudar a convergência é mais complicada do que a utilizada em 11.2.4, mas é possível chegar à seguinte relação, para o erro de discretização,

$$|\epsilon_i^j| \leq \frac{1}{24}(1+r)(\Delta x)^2 M$$

em que $r = \frac{(\Delta x)^2}{(\Delta y)^2}$, M é o majorante das quartas derivadas de u em ordem a x e y e

$$\epsilon_i^j = u(x_i, y_j) - u_i^j.$$

Como, quando $\Delta x \to 0$ também $|\epsilon_i^j| \to 0$, a solução da equação diferença (11.36) é convergente.

11.5 Problemas

1. Calcule a solução da equação

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2},$$

sujeita às condições de fronteira $u(0,t) = u(1,t) = 0$ e à condição inicial $u(x,0) = sen(\pi x)$, até atingir $t = 0.02$.

   a) Use $\Delta x = 0.1$, $\Delta t = 0.005$ e um esquema explícito.
   Repita com $\Delta x = 0.1$ e $\Delta t = 0.01$.

   b) Para os valores $\Delta x = 0.1$ e $\Delta t = 0.01$ use um esquema implícito.

   Compare os valores obtidos com a solução exacta $u(x,t) = exp(-\pi^2 t) sen(\pi x)$.

2. Considere a equação

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} \quad \text{com} \quad 0 \leq x \leq 1$$

sujeita às seguintes condições:

   i) $u(0,t) = u(1,t) = 0, t \geq 0$

   ii) $u(x,0) = 2x, 0 \leq x \leq \frac{1}{2}$

iii) $u(x,0) = 2(1-x), \frac{1}{2} \leq x \leq 1$

Determine, usando um esquema implícito com $r = 1$ e $\Delta x = 0.1$, o valor de $u(0.6,\ 0.01)$.

3. Calcule uma solução da equação diferencial $\frac{\partial^2 u}{\partial t^2} = \frac{\partial^2 u}{\partial x^2}$, para $0 \leq x \leq 1, t \geq 0$, sujeita a $u(x,0) = \frac{1}{2}x(1-x)$, $\frac{\partial u(x,0)}{\partial t} = 0, 0 \leq x \leq 1$ e $u(0,t) = u(1,t) = 0$ para $t \geq 0$. Use $r = 1$ e repita com $r = 0.5$, tomando $\Delta t = 0.25$.

4. Calcule uma solução aproximada para a equação de Laplace $\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$, para $0 \leq x, y \leq 1$, sujeita a $u(0,y) = 0, 0 \leq y \leq 1$; $u(1,y) = 0, 0 \leq y \leq 1$; $u(x,0) = sen^2(\pi x)$ para $0 \leq x \leq 1$ e $u(x,1) = 0$ para $0 \leq x \leq 1$. Tome $\Delta x = \Delta y = \Delta = 0.25$.

5. A distribuição da temperatura $u(x,y)$ para um fluxo de calor a duas dimensões, num material cuja condutividade de calor é constante, satisfaz a equação de Laplace:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$$

Usando uma malha com espaçamento constante entre pontos igual a 1 ($\Delta = 1$), calcule as temperaturas nos pontos interiores do quadrado definido por: $0 \leq x \leq 3$, $0 \leq y \leq 3$.

Considere

$$u(0,y) = u(x,0) = 0$$

$$u(x,3) = x^3$$

e

$$u(3,y) = 9y.$$

Bibliografia

1. P.R. Adby e M.A.H. Dempster - *Introduction to Optimization Methods*, Chapman and Hall, 1974.

2. M.S. Bazaraa, H.D. Sherali e C.M. Shetty - *Nonlinear Programming. Theory and Algorithms*, 2^{nd} Ed., John Wiley & Sons, 1993.

3. W. Blum, M. Niss e I. Huntley (Eds.) - *Modelling, Applications and Applied Problem Solving. Teaching Mathematics in a Real Context*, Ellis Horwood Ltd., England, 1989.

4. S.C. Chapra e R.P. Canale - *Numerical Methods for Engineers with Personal Computer Applications*, McGraw-Hill Inc., 1985.

5. P.G. Ciarlet - *Introduction to Numerical Linear Algebra and Optimization*, Cambridge University Press, 1989.

6. S.D. Conte e C. deBoor - *Elementary Numerical Analysis: An Algorithmic Approach*, 3^{rd} Edition, McGraw-Hill, 1980.

7. M.G. Cox - *A Data Fitting Package for the Non-Specialist User*, em D.J. Evans (Ed.), Software for Numerical Mathematics, Academic Press, 1974.

8. H.P. Crowder, R.S. Dembo e J.M. Mulvey - *On Reporting Computational Experiments with Mathematical Software*, ACM Transactions on Mathematical Software, 5, 193-203, 1979.

9. G. Dahlquist e A. Björck - *Numerical Methods*, Prentice-Hall, Inc., 1974.

10. J.W. Daniel e R.E. Moore - *Computation and Theory in Ordinary Differential Equations*, W.H. Freeman and Company, 1970.

11. J.E. Dennis Jr. e R.B. Schnabel - *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice-Hall, 1983.

12. L.Eldén e L. Wittmeyer-Koch - *Numerical Analysis. An Introduction*, Academic Press, Inc, 1990.

13. A.V. Fiacco e G.P. McCormick - *Nonlinear Programming. Sequential Unconstrained Minimization Techniques*, SIAM, 1990. Oxford University Press, 1964.

14. L. Fox - *An Introduction to Numerical Linear Algebra*, Oxford University Press, 1964.

15. L. Fox e D.F. Mayers - *Computing Methods for Scientists and Engineers*, Clarendon Press, 1968.

16. R.L. Fox - *Optimization Methods for Engineering Design*, Addison-Wesley, 1973.

17. A.C. Freitas - *Introdução à Análise Numérica*, Estudos Gerais Universitários de Moçambique, 1968.

18. G.H. Golub e D.P. O'Leary - *Some History of the Conjugate Gradient and Lanczos Algorithms: 1948 - 1976*, SIAM Review, 31, No. 1, 50 - 102, 1989.

19. P.E. Gill e W. Murray - *Algorithms for the Solution of the Non-Linear Least-Squares Problem*, NPL Report NAC 71, 1976.

20. P.E. Gill, W. Murray e M.H. Wright - *Practical Optimization*, Academic Press, 1981.

21. I. Gladwell e D.K. Sayers (Eds.) - *Computational Techniques for Ordinary Differential Equations*, Academic Press, 1980.

22. H.H. Goldstine - *A History of Numerical Analysis From the* 16^{th} *Through the* 19^{th} *Century*, Springer-Verlag, 1977.

23. G. Hämmerlin e K.-H. Hoffmann - *Numerical Mathematics*, Springer-Verlag, 1991.

24. G.H. Hostetter, M.S. Santina e P.D'Carpio-Montalvo - *Analytical, Numerical, and Computational Methods for Science and Engineering*, Prentice-Hall International Editions, 1991.

25. A.F. Humes, I. de Melo, L Yoshida e W. Martins, *Noções de Cálculo Numérico*, McGraw-Hill, 1984.

26. E. Isaacson e H.B. Keller - *Analysis of Numerical Methods*, John Wiley & Sons, Inc., 1966.

27. R.H.F. Jackson, P.T. Boggs, S.G. Nash e S. Powell - *Guidelines for Reporting Results of Computational Experiments: Report of the Ad Hoc Committee*, Mathematical Programming, 49, 413-425, 1991.

28. L.W. Johnson e R.D. Riess - *Numerical Analysis*, 2^{nd} Edition, Addison-Wesley Publishing Company, 1982.

29. R.L. Johnston - *Numerical Methods. A Software Approach*, John Wiley & Sons, 1982.

30. D. Kahaner, C. Moler e S. Nash - *Numerical Methods and Software*, Prentice-Hall, 1989.

31. G.J. Lastman e N.K. Sinha - *Microcomputers-Based Numerical Methods For Science and Engineering*, Holt, Rinehart and Winston, Inc. Int. Ed.,1989.

32. J.K. Lenstra, A.H.G. Rinnooy Kan e A. Schrijver (Eds.) - *History of Mathematical Programming. A Collection of Personal Reminiscences*, North-Holland, 1991.

33. D. McCracken e W.S. Dorn - *Numerical Methods and Fortran Programming*, John Wiley & Sons, 1964.

34. W. Murray (Ed.) - *Numerical Methods for Unconstrained Optimization*, Academic Press, 1972.

35. S. Nakamura - *Applied Numerical Methods with Software*, Prentice-Hall International, Inc., 1991.

36. J.M. Ortega e W.C. Rheinboldt - *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, 1970.

37. H. Pina - *Métodos Numéricos*, McGraw-Hill, 1995.

38. M.J.D. Powell - *Approximation Theory and Methods*, Cambridge University Press, 1981.

39. A. Ralston e P. Rabinowitz - *A First Course in Numerical Analysis*, McGraw-Hill, 1978.

40. S.S. Rao - *Optimization. Theory and Applications*, Wiley Eastern Lim., 1979.

41. J.R. Rice - *Methodology for the Algorithm Selection Problem*, em Fosdick (Ed.), Performance Evaluation of Numerical Software, North Holland, 1979.

42. J.R. Rice - *Numerical Methods, Software, and Analysis*, McGraw-Hill Inc., 1983.

43. F. Scheid - *Análise Numérica*, 2^a Edição, McGraw-Hill, 1991.

44. L.F. Shampine e M.K. Gordon - *Computer Solution of Ordinary Differential Equations: The Initial Value Problem*, Freeman, 1975.

45. S. Smale - *Some Remarks on the Foundations of Numerical Analysis*, SIAM Review, 32, No. 2, 211 - 220, 1990.

46. G.D. Smith - *Numerical Solution of Partial Differential Equations*, Oxford University Press, 1975.

47. W.H. Swann - *Direct Search Methods*, em W. Murray (Ed.), Numerical Methods for Unconstrained Optimization, Academic Press, 1972.

48. J.F. Traub - *Theory of Optimal Algorithms*, em D.J. Evans (Ed.), Software for Numerical Mathematics, Academic Press, 1974.

49. M.R. Valença - *Métodos Numéricos*, Instituto Nacional de Investigação Científica, 1988.

50. M.R. Valença - *Análise Numérica* - Universidade Aberta, 1996.

51. J.S. Vandergraff - *Introduction to Numerical Computation*, Academic Press, 1978.

52. R.S. Varga -*Matrix Iterative Analysis*, Prentice-Hall, Inc., 1962.

53. J.H. Wilkinson - *Rounding Errors in Algebraic Processes*, Prentice-Hall, Inc., 1963.

54. J.H. Wilkinson - *The Algebraic Eigenvalue Problem*, Clarendon Press, 1965.

55. M.A. Wolfe - *Numerical Methods for Unconstrained Optimization*, Van Nostrand Reinhold, 1978.

56. D.M. Young e R.T. Gregory - *A Survey of Numerical Mathematics* Vol. 1, Addison-Wesley Publ., 1972.

Índice dos algoritmos que correspondem às rotinas numéricas