MÉTODOS NUMÉRICOS

GUIA PRÁTICO

Carlos Lemos

Heitor Pina

INSTITUTO SUPERIOR TÉCNICO

2001

Conteúdo

1

Álgebra Linear: aspectos computacionais

Neste capítulo vamos abordar alguns aspectos práticos da Álgebra Linear Numérica, nomeadamente as questões relativas ao armazenamento e processamento de matrizes de grande dimensão, com vista a tirar partido da estrutura especial que, porventura, possam ter, e, deste modo, poupar memória e tempo de cálculo.

1.1 Introdução

Consideraremos apenas matrizes reais com m linhas e n colunas, i.e., $\mathbf{A} \in \mathbb{R}^{m \times n}$, já que o caso complexo não apresenta qualquer dificuldade adicional. O espaço em memória necessário para armazenar esta matriz é igual a mn, ou seja, a *complexidade espacial* de um problema que envolva uma matriz deste tipo é $C_s = \mathcal{O}(mn)$. Em aplicações práticas, m e n tendem a assumir valores elevados, da ordem das centenas e, mesmo, milhares, pelo que as matrizes com que nos defrontamos são, computacionalmente falando, objectos grandes.

1.1.1 Memória

Um aspecto de enorme importância para uma programação eficiente, especialmente relevante quando se recorre intensivamente a matrizes, é o da organização da memória dos computadores.

Em primeiro lugar é preciso ter em conta que a memória dos computadores é constituída por uma *hierarquia* de dispositivos com características diferentes:

- a *memória central*, que inclui a memória conhecida pela sigla RAM (*Random Access Memory*) e outras como sejam as memórias tampão (*caches*, etc.): rápida, e por isso cara, logo, geralmente escassa;

- *memórias periféricas* (discos, bandas magnéticas, etc.): mais lentas que a memória central, mais baratas e, por isso, de maior volume.

Os projectistas de computadores cedo tiveram que enfrentar o problema decorrente de muitos dos programas (código e dados) possuirem uma dimensão maior que a disponível na memória central. Para resolver esta dificuldade foram inventadas basicamente duas técnicas: a *segmentação* de programas e a *memória virtual*.

Segmentação

A segmentação consiste na divisão, pelo programador, do programa em unidades coerentes (grupos de subrotinas ou procedimentos), cada uma delas podendo caber na memória central. O programa e respectivos dados estão inicialmente residentes em memória periférica. O sistema operativo inicia a execução do programa carregando para memória central um segmento especial, o *núcleo* ou *programa principal*, o qual fica, a partir deste momento, responsável por gerir o carregamento, da memória periférica para a memória central e desta para memória periférica, dos restantes segmentos. Ou seja, a forma de segmentar o programa (organização dos segmentos, sua dimensão, etc.) e a gestão dos segmentos durante a execução ficam a cargo do programador.

Trata-se de uma tarefa altamente dependente do computador utilizado, ingrata para o programador que se vê a braços com a necessidade de "partir" o seu programa, muitas vezes de forma pouco lógica, e de ter que dispender tempo e energia com este tipo de pormenores em detrimento da sua tarefa principal.

Memória virtual

Para obviar aos inconvenientes da segmentação, foi desenvolvida uma alternativa que consiste essencialmente em efectuar uma segmentação automática, transparente para o programador, e que hoje é praticamente universal. Este automatismo é justificado pelo *princípio da localidade* que consiste na regra prática (não é um teorema!, ver PATTERSON and HENNESSY (1996)):

Regra dos 90%/10%**:** um programa gasta 90% do seu tempo na execução de 10% do seu código, i.e., geralmente existem módulos que são computacionalmente muito mais intensivos que os restantes.

Este princípio da localidade desdobra-se em dois *sub-princípios*:

Localidade temporal: se um elemento (instrução ou variável) for referenciado num dado momento, tenderá a ser referenciado em momentos imediatamente próximos;

Localidade espacial: se um elemento for referenciado, os elementos seus vizinhos em memória tenderão a ser referenciados em momentos imediatamente próximos.

Se estes princípios são verdadeiros em geral, são-no ainda mais quando o programa faz uso intenso de vectores e matrizes. De facto, a Álgebra Linear recorre a operações com carácter *sistemático* e *repetitivo* sobre os elementos dos vectores e matrizes. Dito por outras palavras: um programa (código e dados) não precisa estar todo em memória para poder funcionar; basta que residam em memória os elementos vizinhos (no espaço e no tempo).

A chave para a materialização desta ideia passa, entre outras, pela introdução do conceito de *memória virtual* (em contraponto à memória *real*). A memória virtual funciona do seguinte modo:

1. O compilador divide um programa, qualquer que seja o respectivo tamanho, em segmentos de comprimento *fixo* (valores típicos entre $2^9 = 512$ e $2^{13} = 8192$ Bytes); estes segmentos são conhecidos pela designação sugestiva de *páginas* (o programa é o "livro"), às quais é atribuído um "número de página" ou endereço;

2. O conjunto das páginas que constituem o programa reside em disco, e, portanto, o limite do tamanho de um programa é determinado pela capacidade do disco e não pela capacidade da memória central;

3. A execução do programa inicia-se com o carregamento para memória central de tantas páginas quantas as que cabem na parte da memória central alocada ao programa;

4. Quando o programa, durante a sua execução, invoca uma página não residente em memória central, i.e., quando se verifica uma *falta* de página, o sistema operativo desencadeia a transferência de uma página que não está a ser precisa para a memória periférica e a leitura da página que está a ser precisa da memória periférica para a memória central. Este processo de leitura/escrita é conhecido por *paginação* e um problema interessante é o de saber qual a página que deve ser enviada para a memória periférica (a mais "antiga", a menos "usada"?) para dar lugar à nova página.

A memória virtual consiste assim no espaço que o sistema operativo/compilador permite endereçar o qual está, como se disse, limitado fisicamente pela capacidade da memória periférica. Durante a execução do programa, as páginas da memória virtual são mapeadas na memória real.

É conveniente ter em atenção as escalas de tempo envolvidas: uma operação aritmética em ponto flutuante pode levar 10^{-6} segundos enquanto uma falta de página pode requerer 10^{-3} segundos, ambos valores típicos. Quer isto dizer que o tempo de execução de um programa pode ser determinado, não pelas operações aritmética em ponto flutuante, mas pela excessiva paginação que provoca. Compete ao programador, alertado para esta eventualidade, evitar paginação desnecessária. A regra para tal resume-se simplesmente nisto: aceder ou invocar, tanto quanto for possível, páginas contíguas na memória virtual

(portanto, com alta probabilidade de estarem simultaneamente residentes na memória real) e evitar páginas "distantes" (i.e., com grande probabilidade de não residirem em simultâneo na memória central).

Exercício 1.1.1 *Considere o caso de um computador com páginas de* 1024 *Bytes e números reais ocupando palavras de* 8 *Bytes. a) Quantas multiplicações é que o Algoritmo 1.3.2 executa antes de "saltar"de página? Estude o caso matrizes quadradas de ordem* 10, 10^2 *e* 10^3*. b) E se trocarmos a ordem dos ciclos como se faz no Algoritmo 1.3.3?*

1.1.2 Armazenamento de matrizes

Todas as linguagens de programação de alto nível oferecem estruturas de dados aptas a representar matrizes e a operar directamente com os respectivos elementos. Tal requer geralmente uma declaração do tipo

dimension $\mathbf{A}(30, 20)$

que permite ao compilador:

- saber que a variável designada por $\mathbf{A}$ é uma matriz com 30 linhas e 20 colunas e reservar o espaço apropriado ($30 \times 20 = 600$ palavras);
- interpretar correctamente $a(i, j)$ (ou $a[i, j], \ldots$, consoante a linguagem de programação) como referências às variáveis matemáticas a_{ij}.

Contudo, esta facilidade oferecida a nível dos compiladores, nunca nos deve fazer esquecer que a memória de um computador é essencialmente *linear*, i.e., que as variáveis são armazenadas sequencialmente, umas a seguir às outras: não há matrizes, com linhas e colunas, na memória de um computador! As matrizes são, em computador, ficções criadas pelo compilador para benefício do programador.

Exemplificando: seja $\mathbf{A} \in \mathbb{R}^{3\times 2}$. Então, dependendo da linguagem de programação, os 6 elementos de $\mathbf{A}$ são armazenados *por colunas* (é o caso do Fortran) da seguinte maneira:

$$\left| a_{11} \right| a_{21} \left| a_{31} \right| a_{12} \left| a_{22} \right| a_{32} \left| \right. \tag{1.1.1}$$

ou *por linhas* (é o caso do Pascal):

$$\left| a_{11} \right| a_{12} \left| a_{21} \right| a_{22} \left| a_{31} \right| a_{32} \left| \right. \tag{1.1.2}$$

O que se passa quando nos referimos ao elemento a_{ij} (aliás, $a(i, j)$, $a[i, j]$, ...), de $\mathbf{A} \in \mathbb{R}^{m\times n}$? Simplesmente isto: o compilador toma nota das dimensões m e n e reserva $N = mn$ palavras a partir do *endereço base* do primeiro elemento da matriz, a_{11}. O

endereço do elemento a_{ij} é calculado somando ao endereço base um *deslocamento*. É um exercício simples ver que ao índice (i, j) corresponde o deslocamento k dado por

$$(i,j) \mapsto k = \begin{cases} (j-1)m + i - 1 & \text{(armazenamento por colunas)} \\ (i-1)n + j - 1 & \text{(armazenamento por linhas)} \end{cases} \tag{1.1.3}$$

e é este *mapeamento* do índice bidimensional (i, j) num índice unidimensional k que o compilador efectua de forma transparente para o programador. Notemos que estas operações de indexação, não obstante envolverem apenas inteiros, não são totalmente de ignorar no cômputo do tempo de cálculo total.

Uma consequência imediata do que se acabou de dizer é a de que todo e qualquer desvio relativamente a este esquema de armazenamento implica termos de desistir desta facilidade dos compiladores. Dito por outras palavras: a indexação passa a ser da responsabilidade do programador, tendo em conta a estrutura de dados adoptada para armazenar as matrizes. Há um preço a pagar: a correspondência entre a descrição matemática do algoritmo e o programa torna-se menos evidente.

Exercício 1.1.2 *Mostre que no que diz respeito a vectores $\mathbf{x} \in \mathbb{R}^n$ não há, do ponto de vista de armazenamento e indexação, qualquer diferença entre vectores linha $\mathbf{A} \in \mathbb{R}^{1\times n}$ e vectores coluna $\mathbf{A} \in \mathbb{R}^{n\times 1}$.*

1.1.3 Estrutura e esparsidade

Frequentemente, as matrizes possuem propriedades que tornam possíveis esquemas de armazenamento menos exigentes. Uma delas tem a ver com a *esparsidade*.

Uma matriz diz-se *esparsa* se possuir muitos elementos nulos. Como se vê, trata-se de uma noção difusa já que o significado de "muitos elementos" não é claro. De facto, é o contexto que dita quando é que uma matriz deve ser considerada esparsa ou se, pelo contrário, deve ser considerada *densa*.

Basicamente, as características que devemos considerar ao desenhar um esquema alternativo de armazenamento são as seguintes:

Estrutura: a matriz possui uma estrutura especial conhecida *a priori*, por exemplo, a matriz é simétrica.

Esparsidade: a matriz é *esparsa*, i.e., possui uma percentagem elevada de zeros que, eventualmente, não são explicitamente necessários, querendo com isto dizer que podemos tê-los em conta por via de uma programação apropriada. É ainda conveniente verificar se esta esparsidade é:

organizada, ou seja, os zeros estão localizados em posições que formam um padrão conhecido *a priori*;

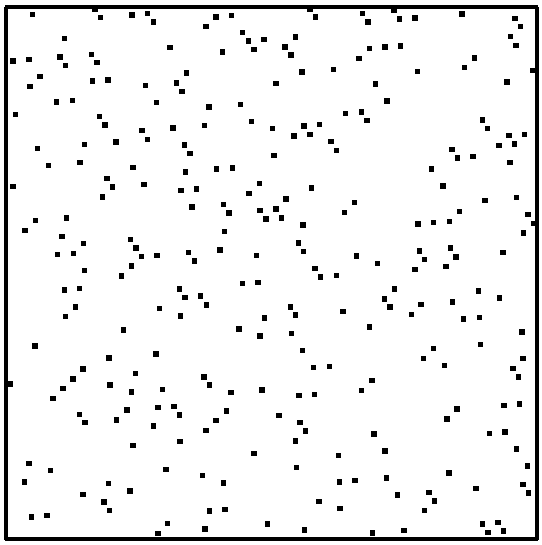


Figura 1.1.1: Matriz com esparsidade desorganizada: os quadrados pretos indicam elementos não nulos

desorganizada, no caso contrário.

e ainda, se é:

estática, i.e., o padrão de esparsidade não é alterado durante o processamento da matriz;

dinâmica, se este padrão é modificado.

As Figuras 1.1.1 e 1.1.2 mostram matrizes com diferentes padrões de esparsidade e os exemplos seguintes ilustram também várias situações correntes.[1]

Exemplo 1.1.1 *Seja* $\mathbf{A} \in \mathbb{R}^{n\times n}$ *uma matriz tridiagonal. Em que casos a podemos considerar?*

Trata-se obviamente de uma matriz com estrutura especial, induzida pela tridiagonalidade, tem um padrão de esparsidade organizado pois só os elementos da diagonal principal e das codiagonais superior e inferior é que poderão ser diferentes de zero. (Nota: eventuais zeros nestas diagonais não são relevantes). Para prosseguir, é preciso saber que operações é que irão ser efectuadas com a matriz. Se a matriz se destinar a ser multiplicada por escalares ou adicionada a outras matrizes tridiagonais, então é fácil ver que um esquema de armazenamento estático é suficiente; de facto, três vectores de dimensão n chegam para representar $\mathbf{A}$, um para a diagonal principal e outros dois para as codiagonais. Mas, se pelo contrário, se pretende efectuar a condensação da matriz com troca de linhas por escolha de pivot, então torna-se necessária um quarto vector para acolher os elementos que mudam de posição.

Em qualquer caso, estamos perante uma complexidade espacial de $\mathcal{O}(n)$ em vez de $\mathcal{O}(n^2)$ como no caso geral o que constitui uma grande economia. ∎

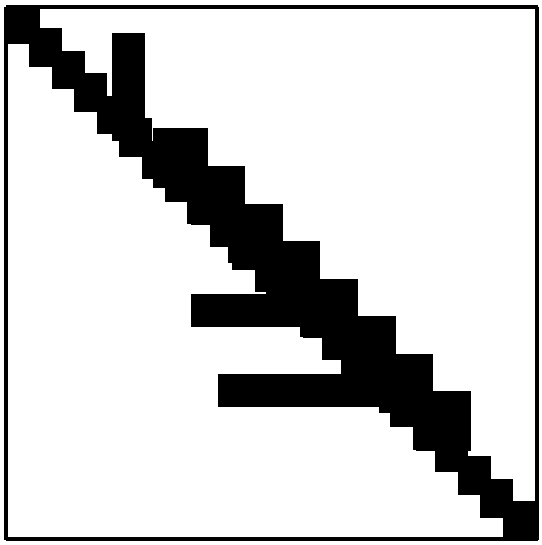


Figura 1.1.2: Matriz com esparsidade organizada: a zona preta indica a presença de elementos não nulos

1.1.4 Matrizes e grafos

Em muitas situações torna-se vantajoso recorrer a diagramas como os representados na Figura 1.1.3. Por exemplo, os círculos podem representar cidades e os arcos as estradas que ligam essas cidades. Assim, no primeiro caso, o diagrama da esquerda indica-nos a existência de uma estrada que liga a cidade v_1 à cidade v_2 e outra estrada (não tem que ser a mesma) que liga a cidade v_2 à cidade v_1, *idem* para as cidades v_2 e v_3, de uma estrada que liga a cidade v_3 à cidade v_4, etc. No diagrama da direita, sem setas, presume-se que as ligações são sempre bidireccionais, como sucede se os círculos indicarem centrais de comutação e os arcos cabos de telecomunicações, por exemplo.

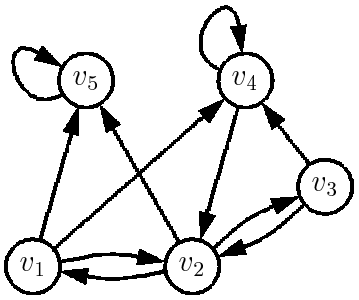
O objecto matemático que traduz estes diagramas é designado por *grafo* G e é definido por dois conjuntos: um conjunto de *vértices* $V = \{v_1, \ldots, v_n\}$ e um conjunto de *arcos* A em que cada arco é constituído por um par de vértices (v_i, v_j). Se este pares forem *ordenados* diz-se que o grafo é *orientado*, caso contrário diz-se que o grafo é *não orientado*. Neste caso, os arcos tomam a designação de *arestas*.

NOTA. Neste texto apenas trataremos de *grafos finitos*, i.e., com um número de vértices, denotado por $|V|$, e um número de arcos, denotado por $|A|$, ambos finitos. ∎

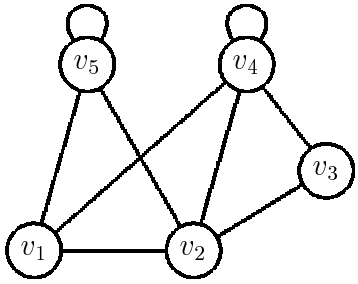
Assim, na Figura 1.1.3, o grafo orientado é dado por:

$$\begin{aligned} V &= \{v_1, v_2, v_3, v_4, v_5\} \\ A &= \{(v_1, v_2), (v_1, v_4), (v_1, v_5), (v_2, v_1), (v_2, v_3), (v_2, v_5), (v_3, v_4), (v_4, v_4), (v_5, v_5)\} \end{aligned}$$

Figura 1.1.3: Exemplos de grafos



Orientado



Não orientado

e o grafo não orientado por:

$$\begin{aligned} V &= \{v_1, v_2, v_3, v_4, v_5\} \\ A &= \{(v_1, v_2), (v_1, v_4), (v_1, v_5), (v_2, v_3), (v_2, v_5), (v_3, v_4), (v_4, v_4), (v_5, v_5)\} \end{aligned}$$

Num grafo orientado, dado um arco (v_i, v_j) diz-se que v_i é o vértice *inicial* e v_j o vértice *terminal* desse arco. O arco (v_i, v_j) diz-se *incidente para o exterior* em v_i e *incidente para o interior* em v_j.

Dois arcos distintos são *adjacentes* se possuírem um vértice (inicial ou terminal) comum. Identicamente, dois vértices distintos dizem-se *adjacentes* se existir (pelo menos) um arco entre eles.

A *matriz de adjacência* $\mathbf{G} \in \mathbb{R}^{n \times n}$ de um grafo G é a matriz cujos elementos são definidos por

$$g_{\pi(i)\pi(j)} = \text{número de arcos } (v_i, v_j) \in A$$

em que π designa uma permutação dos inteiros $\{1, 2, \ldots, n\}$. Em palavras singelas: π representa a *rotulação* dos vértices com inteiros de 1 a n ou, se quisermos, a correspondência entre vértices e linhas e colunas da matriz de adjacência. A forma mais simples é tomar $\pi(i) = i$, mas não tem que ser assim e há situações em que convém proceder de outra forma como se verá mais adiante.

O *grau exterior* de um vértice v é o número de arcos com extremidade inicial em v e denotar-se-á por $d^+(v)$; analogamente, *grau interior* de um vértice v é o número de arcos com extremidade terminal em v e denotar-se-á por $d^-(v)$. O *grau* de um vértice v é simplesmente $d(v) = d^+(v) + d^-(v)$ e é igual, portanto, ao número de arcos incidentes em v. O vértice v diz-se que é uma *entrada* se $d^+(v) > 0$ e $d^-(v) = 0$, uma *saída* se $d^-(v) > 0$ e $d^+(v) = 0$ e *isolado* se $d(v) = 0$.

Como cada arco é exactamente incidente para o exterior num (e num só) vértice e exactamente incidente para o interior num (e num só) vértice, não oferece dificuldade estabelecer que

$$\sum_{v \in V} d^+(v) = \sum_{v \in V} d^-(v) = |A|$$

e, por conseguinte,

$$\sum_{v \in V} d(v) = 2|A|$$

Estas relações podem ser expressas em termos dos coeficientes da matriz de adjacência. De facto, não oferece qualquer dificuldade estabelecer que

$$\begin{aligned} d^+(v) &= \sum_{j=1}^{n} g_{ij} \\ d^-(v) &= \sum_{i=1}^{n} g_{ij} \end{aligned}$$

ou seja, o grau exterior pode obter-se por soma da linha de $\mathbf{G}$ e o grau interior por soma da coluna de $\mathbf{G}$ associada ao vértice.

Exemplo 1.1.2 *Obter, para os grafos da Figura 1.1.3, as matrizes de adjacência e os graus dos vértices.*

As matrizes de adjacência são, respectivamente:

$$\mathbf{G} = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \qquad \mathbf{G} = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

Os graus calculam-se simplesmente a partir destas matrizes, vindo respectivamente

	v_1	v_2	v_3	v_4	v_5
d^+	3	3	2	2	1
d^-	1	3	1	3	3
d	4	6	3	5	4

	v_1	v_2	v_3	v_4	v_5
d	3	4	2	4	3

Notemos que para um grafo não orientado não faz sentido falar em graus exterior e interior mas sim, simplesmente, em grau. ∎

A cada grafo corresponde pois uma matriz de adjacência e, dada uma matriz de adjacência, é sempre possível reconstruir o respectivo grafo. O aspecto que nos interessa mais no contexto da Álgebra Linear Numérica é o de que a uma matriz $\mathbf{A} \in \mathbb{R}^{n \times n}$ pode sempre associar-se uma matriz de adjacência, e portanto, um grafo fazendo

$$g_{ij} = \begin{cases} 1 & \text{se } a_{ij} \neq 0, \\ 0 & \text{se } a_{ij} = 0. \end{cases}$$

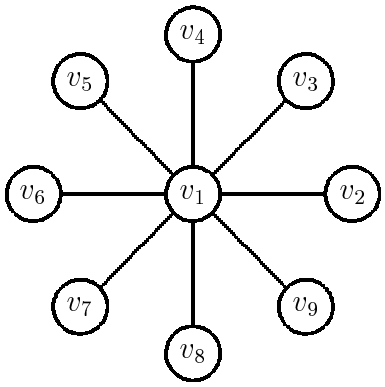


Figura 1.1.4: Exemplo de um grafo em estrela

ou seja, criando um vértice por cada linha (ou coluna) e um arco por cada coeficiente não nulo de $\mathbf{A}$. A matriz de adjacência, ou o grafo, reflectem a estrutura e o padrão de esparsidade de $\mathbf{A}$.

A cada arco (v_i, v_j) pode associar-se um valor a_{ij}; por exemplo, o comprimento da estrada que liga as cidades, ou o custo das respectivas portagens, ou a sua capacidade de tráfego. Assim, a cada grafo pode corresponder uma matriz $\mathbf{A}$ e, reciprocamente, a cada matriz pode associar-se um grafo. Para os casos da Figura 1.1.3, temos as seguintes matrizes:

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & 0 & a_{14} & a_{15} \\ a_{21} & a_{22} & a_{23} & 0 & a_{25} \\ 0 & a_{32} & a_{33} & a_{34} & 0 \\ 0 & 0 & 0 & a_{44} & 0 \\ 0 & 0 & 0 & 0 & a_{55} \end{pmatrix}, \qquad \mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & 0 & a_{14} & a_{15} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\ 0 & a_{32} & a_{33} & a_{34} & 0 \\ a_{41} & a_{42} & a_{43} & a_{44} & 0 \\ a_{51} & a_{52} & 0 & 0 & a_{55} \end{pmatrix}$$

Observemos que o padrão de esparsidade da matriz $\mathbf{A}$ reflecte precisamente a conectividade do respectivo grafo e que um grafo não orientado corresponde a uma matriz simétrica e vice-versa.

Exercício 1.1.3 *Determinar o grafo de uma matriz tridiagonal: a) não simétrica; b) simétrica.*

Exercício 1.1.4 *Determinar, para o grafo da Figura 1.1.4, as matrizes de adjacência quando se adoptam as seguintes numerações para os vértices: a)* $\pi(i) = i$*; b)* $\pi(i) = 10 - i$*. Do ponto de vista da aplicação do método de Gauss, qual é a mais favorável?*

1.2 Armazenamento de matrizes especiais

Aqui vamos ter necessidade de distinguir a matriz $\mathbf{A}$ da estrutura de dados, que denotaremos simplesmente por A, que vai constituir a sua representação interna, i.e., que

vai armazenar os elementos relevantes de $\mathbf{A}$. Dito por outras palavras, $\mathbf{A}$ é um objecto matemático (uma matriz) e A um objecto de programa (geralmente uma tabela) que guarda os elementos de $\mathbf{A}$ "que interessam". Torna-se evidente que é indispensável construir, conjuntamente com A, a correspondência entre os elementos de $\mathbf{A}$ "que interessam" e A, ou, o *mapeamento* da parte "interessante" de $\mathbf{A}$ em A.

1.2.1 Matrizes simétricas

Um dos casos mais frequentes nas aplicações é o das matrizes simétricas. Se o processamento subsequente não destruir a simetria, podemos assim trabalhar apenas com um dos triângulos (superior ou inferior) da matriz.

Suponhamos então que $\mathbf{A} \in \mathbb{R}^{n\times n}$ é simétrica e desejamos armazenar apenas o seu triângulo superior. Se optarmos por um *armazenamento por colunas*, então os elementos a armazenar em A são, pela ordem indicada,

$$A = \left| a_{11} \right| a_{12} \left| a_{22} \right| a_{13} \left| a_{23} \right| a_{33} \left| \cdots \right| a_{1n} \left| a_{2n} \right| \cdots \left| a_{nn} \right| \tag{1.2.1}$$

A tabela A tem, deste modo, $N = n(n+1)/2$ elementos. Neste caso, o mapeamento do índice bidimensional (i,j) de $\mathbf{A}$ num índice unidimensional k de A (leia-se $A(k) = a_{ij}$) é dado por

$$(i,j) \mapsto k = \frac{j(j-1)}{2} + i, \qquad j \geq i \tag{1.2.2}$$

Se optarmos por um *armazenamento por linhas*, i.e.,

$$A = \left| a_{11} \right| a_{12} \left| \cdots \right| a_{1n} \left| a_{22} \right| a_{23} \left| \cdots \right| a_{2n} \left| \cdots \right| a_{nn} \left| \right. \tag{1.2.3}$$

não oferece dificuldade ver que, neste caso,

$$(i,j) \mapsto k = \frac{(2n+2-i)(i-1)}{2} + j - i + 1, \qquad j \geq i \tag{1.2.4}$$

Exercício 1.2.1 *Quais são as expressões equivalentes a* (1.2.2) *e* (1.2.4) *no caso de se pretender armazenar o triângulo inferior?*

NOTA. Tudo o que se acabou de dizer tem aplicação directa ao caso de se pretender armazenar apenas o triângulo não nulo de matrizes triangulares. ∎

1.2.2 Matrizes esparsas

Vamos expor nesta secção os esquemas mais frequentes para armazenar matrizes esparsas. Há que ter em atenção que o respectivo comportamento quer em termos estáticos, i.e., a estrutura de esparsidade não se altera durante o processamento, quer em termos dinâmicos, i.e., esta estrutura altera-se, é muito diferente. Assim, um bom esquema estático pode ser

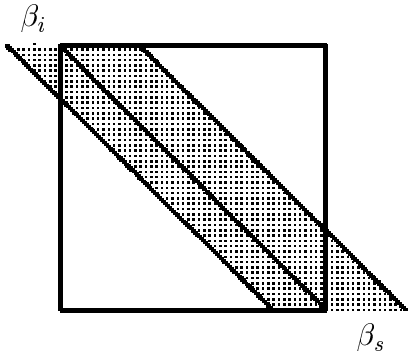


Figura 1.2.1: Matriz banda

demasiado inflexível ao não permitir, por exemplo, a inclusão de elementos inicialmente nulos mas que deixaram de o ser durante o processamento. É o que acontece durante a condensação de uma matriz pelo método de Gauss em que elementos originalmente nulos deixam de o ser e elementos originalmente não nulos passam a assumir o valor zero.

NOTA. Tudo o que se vai dizer relativamente a matrizes tem aplicação directa ao caso de vectores, com as adaptações óbvias. ∎

Matrizes em banda

Se $\mathbf{A} \in \mathbb{R}^{n\times n}$ possuir β_i codiagonais inferiores e β_s codiagonais superiores, então para armazenar os elementos não nulos desta matriz basta uma tabela rectangular, A, digamos, com n linhas e $\beta_i + \beta_s + 1$ colunas, logo com $N = n(\beta_i + \beta_s + 1)$ elementos (ver a Figura 1.2.1). Aceitamos deste modo um certo desperdício de memória ao guardar também as "pontas" triangulares sem utilidade, o que não tem importância já que este esquema de armazenamento só faz sentido se $\beta_i \ll n$ e $\beta_s \ll n$, e tal traduzir-se portanto numa perda perfeitamente aceitável. O mapeamento (i, j) dos elementos da matriz $\mathbf{A}$ nos elementos (i', j') da tabela A, vem dado por

$$(i, j) \mapsto (i', j') = (i, \beta_i + j - i + 1) \tag{1.2.5}$$

Exercício 1.2.2 *O grafo da Figura 1.2.2 representa uma estrutura, uma rede de comunicações, etc. Qual a melhor maneira de numerar os vértices com vista a obter uma matriz com uma banda tão estreita quanto possível?*

Armazenamento aleatório

Uma forma de armazenar uma matriz esparsa $\mathbf{A} \in \mathbb{R}^{m\times n}$ consiste em guardar os N, digamos, elementos não nulos de $\mathbf{A}$ numa tabela unidimensional A e utilizar duas tabelas unidimensionais de inteiros, i_A e j_A, com a função de *apontadores*, para guardar os respectivos índices das linhas e das colunas. Esta ideia é melhor explicada com um exemplo.

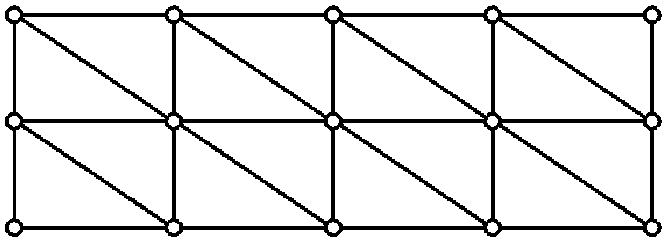


Figura 1.2.2: Grafo de uma estrutura, rede, etc.

Exemplo 1.2.1 *Obter o esquema de armazenamento aleatório para* $\mathbf{A} \in \mathbb{R}^{4\times 5}$ *dada por*

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 2 & 3 & 0 \\ 0 & 0 & 4 & 5 & 0 \\ 6 & 0 & 0 & 0 & 7 \\ 0 & 8 & 0 & 0 & 0 \end{pmatrix}$$

Temos neste caso que

$$\begin{aligned} A &= \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{pmatrix} \\ i_A &= \begin{pmatrix} 1 & 1 & 1 & 2 & 2 & 3 & 3 & 4 \end{pmatrix} \\ j_A &= \begin{pmatrix} 2 & 3 & 4 & 3 & 4 & 1 & 5 & 2 \end{pmatrix} \end{aligned}$$

Isto significa, por exemplo, que $i_A(3) = 1$ e $j_A(3) = 4$, logo $a_{14} = A(3) = 3$. ∎

Com este esquema de armazenamento extremamente flexível é muito fácil acrescentar novos elementos não nulos à matriz $\mathbf{A}$ para o que basta criar as entradas necessárias nas tabelas A e nos apontadores i_A e j_A. Todavia este esquema de armazenamento só é compensador para matrizes muito esparsas conforme o exercício seguinte prova.

Exercício 1.2.3 *Suponha que um inteiro é representado por uma palavra e um real por* r *palavras (habitualmente* $r = 1$, $r = 2$ *ou* $r = 4$*). Determine quão esparsa uma matriz* $\mathbf{A} \in \mathbb{R}^{n\times n}$ *precisa de ser para que o armazenamento aleatório seja menos exigente em memória do que o armazenamento total (o qual requer* rn^2 *palavras).*

O problema inverso, i.e., obter o valor de a_{ij} obriga a determinar k tal que $i_A(k) = i$ e $j_A(k) = j$; se este valor existir, então $a_{ij} = A(k)$, se não $a_{ij} = 0$. Como se vê, é necessário pesquisar nas duas tabelas i_A e j_A, o que pode tornar-se pesado.

Exercício 1.2.4 *Escrever um pseudo programa para somar duas matrizes* $\mathbf{C} = \mathbf{A} + \mathbf{B}$ *com* $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{m\times n}$ *usando armazenagem aleatória (construindo, na passagem, as tabelas auxiliares* i_C *e* j_C*).*

Armazenamento comprimido por linhas

Este esquema é uma variante do anterior em que, para além da tabela principal A onde se guardam os elementos não nulos de $\mathbf{A} \in \mathbb{R}^{m\times n}$ *por linhas*, se utilizam duas tabelas auxiliares: j_A, com o mesmo significado que anteriormente, e ℓ_A, em que $\ell_A(k)$ aponta para o elemento onde começa a linha k. Vejamos um exemplo elucidativo.

Exemplo 1.2.2 *Construir o esquema de armazenamento comprimido por linhas para a matriz do Exemplo 1.2.1.*

Temos neste caso que

$$\begin{array}{rcl} A & = & \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{pmatrix} \\ j_A & = & \begin{pmatrix} 2 & 3 & 4 & 3 & 4 & 1 & 5 & 2 \end{pmatrix} \\ \ell_A & = & \begin{pmatrix} 1 & 4 & 6 & 8 & 8 \end{pmatrix} \end{array}$$

Isto significa, por exemplo, que $j_A(5) = 4$ e $\ell_A(2) \leq 5 < \ell_A(3)$, logo $a_{24} = A(5) = 5$. ∎

NOTA. Costuma utilizar-se o último elemento de ℓ para guardar o número de elementos não nulos de $\mathbf{A}$; daí ℓ ter 5 elementos em vez dos 4 como seria de esperar. ∎

Armazenamento comprimido por colunas

É idêntico ao anterior excepto que a tabela principal A guarda os elementos não nulos de $\mathbf{A} \in \mathbb{R}^{m\times n}$ *por colunas*, se utilizam duas tabelas auxiliares: i_A, com o mesmo significado que anteriormente, e ℓ_A, em que $\ell_A(k)$ aponta para o elemento onde começa a coluna k.

Exemplo 1.2.3 *Construir o esquema de armazenamento comprimido por colunas para a matriz do Exemplo 1.2.1.*

Temos neste caso que

$$\begin{array}{rcl} A & = & \begin{pmatrix} 6 & 1 & 8 & 2 & 4 & 3 & 5 & 7 \end{pmatrix} \\ i_A & = & \begin{pmatrix} 3 & 1 & 4 & 1 & 2 & 1 & 2 & 3 \end{pmatrix} \\ \ell_A & = & \begin{pmatrix} 1 & 2 & 4 & 6 & 8 \end{pmatrix} \end{array}$$

Isto significa, por exemplo, que $i_A(5) = 2$ e $\ell_A(3) \leq 5 < \ell_A(4)$, logo $a_{23} = A(5) = 4$. ∎

NOTA. Costuma utilizar-se o último elemento de ℓ para guardar o número de elementos não nulos de $\mathbf{A}$; daí ℓ ter 5 elementos em vez dos 4 como seria de esperar. ∎

Armazenamento comprimido da envolvente

Em muitas situações em que o armazenamento por banda poderia ser considerado, verifica-se que a "distância" entre o primeiro e último elemento não nulo de cada linha varia fortemente de linha para linha pelo que uma banda "largura constante" seria ineficiente. Ou seja, deve-se armazenar a envolvente dos elementos não nulos da matriz e construir as tabelas auxiliares necessárias (ver a Figura 1.1.2). Notemos que neste esquema, tal como no caso de matrizes em banda, os eventuais elementos não nulos dentro da envolvente são guardados.

Neste esquema são necessárias duas tabelas: a tabela principal A para guardar os elementos relevantes de $\mathbf{A}$ e uma tabela auxiliar ℓ_A que aponta para o início de cada linha (se optarmos por um armazenamento orientado por linhas) ou para o início de cada coluna (se optarmos por um armazenamento orientado por colunas).

Exemplo 1.2.4 *Construir o esquema de armazenamento por envolvente (orientado por linhas) para a matriz do Exemplo 1.2.1.*

Temos neste caso que

$$\begin{aligned} A &= \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 0 & 0 & 0 & 7 & 8 \end{pmatrix} \\ \ell_A &= \begin{pmatrix} 1 & 4 & 6 & 8 & 8 \end{pmatrix} \end{aligned}$$

Isto significa, por exemplo, que $\ell_A(2) \leq 5 < \ell_A(3)$, logo $a_{24} = A(5) = 5$. ∎

1.3 Operações com matrizes

Nesta secção vamos abordar as principais operações da Álgebra Linear Numérica, em particular as que envolvem matrizes. A abordagem algorítmica segue de perto a filosofia da programateca BLAS (*Basic Linear Algebra Subroutines*, DONGARRA et al. (1988)), omitindo, no entanto, muitos detalhes técnicos em favor de uma maior clareza de exposição.

1.3.1 Inicialização

Uma das operações mais frequentes em Álgebra Linear Numérica consiste em *inicializar* um vector ou uma matriz, i.e., fazer todos os respectivos elementos iguais a um certo escalar s, frequentemente $s = 0$. Os Algoritmos 1.3.1 e 1.3.2 levam a cabo precisamente esta tarefa e, dada a sua simplicidade, prescindem de qualquer explicação.

Por razões que serão explicadas a seguir, pode ser vantajoso trocar a ordem dos ciclos em i e j no Algoritmo 1.3.2, obtendo-se o Algoritmo 1.3.3.

Algoritmo 1.3.1 Inicialização de vector

```
proc SV (x, s, n)
  comment: x ∈ R^n
  for i = 1 to n do x_i = s od
end
```

Algoritmo 1.3.2 Inicialização de matriz (por linhas)

```
proc SMR (A, s, m, n)
  comment: A ∈ R^{m×n}
  for i = 1 to m do
      for j = 1 to n do
          a_ij = s
      od
  od
end
```

1.3.2 Transposição

A transposição de matrizes é uma das operações mais simples. O Algoritmo 1.3.4 apresenta o caso de matrizes quadradas em que a transposta pode ser armazenada sobre a matriz original, destruindo-a.

Observemos que nesta versão do algoritmo a matriz $\mathbf{A}$ é acedida por linhas mas a invocação do elemento a_{ji} pode provocar paginação excessiva, tornando uma operação simples, como é a transposição, num processo que pode ser pesado computacionalmente. É por esta razão que a formação *explícita* da transposta não é utilizada nas boas programatecas de Álgebra Linear Numérica[2] e deve ser também evitada em qualquer programa que pretenda ser eficiente.

Exercício 1.3.1 *Em termos de grafos, a que corresponde a transposição de matrizes?*

Exercício 1.3.2 *Que tipo de grafos e matrizes de adjacência estão associados a matrizes anti-simétricas? Dê um exemplo.*

1.3.3 Soma e produto por escalar

Somas e produtos por escalares são operações sem dificuldade de maior.

Exercício 1.3.3 *Tomando o Algoritmo 1.3.4 como modelo, escreva um pseudo programa para a soma de matrizes* $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{m\times n}$.

Exercício 1.3.4 Idem, *para a soma de matrizes simétricas* $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n\times n}$.

Algoritmo 1.3.3 Inicialização de matriz (por colunas)

```
proc SMC (A, s, m, n)
  comment: A ∈ R^{m×n}
  for j = 1 to n do
      for i = 1 to m do
          a_ij = s
      od
  od
end
```

Algoritmo 1.3.4 Transposição de matrizes

```
proc TRANS (A, n)
  comment: A ∈ R^{n×n}
  for i = 1 to n do
      for j = 1 to n do
          t = a_ji                                  (salvar a_ji)
          a_ji = a_ij
          a_ij = t
      od
  od
end
```

Exercício 1.3.5 Idem, *para a multiplicação de uma matriz* $\mathbf{A} \in \mathbb{R}^{m\times n}$ *por um escalar* $s \in \mathbb{R}$.

Exercício 1.3.6 Idem, *para o caso de uma matriz* $\mathbf{A} \in \mathbb{R}^{n\times n}$ *simétrica.*

Uma operação frequente em Álgebra Linear Numérica, nomeadamente na condensação de Gauss, é a soma a um vector do produto de outro vector por um escalar. O Algoritmo 1.3.5 exprime este processo.

NOTA. Esta operação corresponde *grosso modo* à subrotina SAXPY e à tríada ligada (*linked triad*) da programateca BLAS. ∎

Exercício 1.3.7 *Quantos flops são requeridos no Algoritmo 1.3.5?*

Exercício 1.3.8 *Tomando o Algoritmo 1.3.5 como modelo, escreva um pseudo programa para obter* $\mathbf{C} = s\mathbf{A} + \mathbf{B}$ *com* $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{m\times n}$ *e* $s \in \mathbb{R}$.

Exercício 1.3.9 Idem, *para o caso de matrizes* $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n\times n}$ *simétricas.*

Algoritmo 1.3.5 Tríada ligada

proc SXPY $(s, \mathbf{x}, \mathbf{y}, \mathbf{z}, n)$
  comment: $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbb{R}^n, s \in \mathbb{R}, \quad \mathbf{z} = s\mathbf{x} + \mathbf{y}$
  for $i = 1$ to n do
    $z_i = sx_i + y_i$
  od
end

1.3.4 Multiplicação

Como preparativo à multiplicação de matrizes, vamos considerar primeiro a operação de produto interno de vectores e, posteriormente, a de multiplicação de matriz por vector. O produto interno de dois vectores $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ é objecto do Algoritmo 1.3.6 cuja interpretação não oferece qualquer dificuldade.

Algoritmo 1.3.6 Produto interno de vectores

proc DOT $(\mathbf{x}, \mathbf{y}, n)$
  comment: $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, $s = \mathbf{y}^T\mathbf{x}$
  $s = 0$ (acumulador)
  for $i = 1$ to n do
    $s = s + x_i y_i$
  od
end

O Algoritmo 1.3.6 é um exemplo das chamadas operações de nível 1 do BLAS (*level-1 BLAS*), i.e., operações que envolvem apenas vectores. O nível 2 do BLAS (*level-2 BLAS*) corresponde a operações envolvendo matrizes e vectores, como o produto de uma matriz por um vector, e o nível 3 do BLAS (*level-3 BLAS*) diz respeito a operações envolvendo matrizes como, por exemplo, o produto de uma matriz por outra matriz.

Exercício 1.3.10 *Calcule a complexidade temporal, em flops, do Algoritmo 1.3.6 em função de n.*

Consideremos agora uma outra operação frequente, a multiplicção de uma matriz por um vector, i.e., $\mathbf{y} = \mathbf{A}\mathbf{x}$. O Algoritmo 1.3.7 reflecte directamente a respectiva definição matemática.

Neste algoritmo a matriz $\mathbf{A}$ é acedida por linhas. Se o compilador armazena matrizes por colunas então este algoritmo pode ser ineficiente do ponto de vista de paginação. Uma alteração simples, baseada na interpretação de $\mathbf{A}\mathbf{x}$ como a combinação linear das colunas de $\mathbf{A}$ cujos coeficientes são as componentes de $\mathbf{x}$, é apresentada no Algoritmo 1.3.8. De notar a alteração da ordem dos ciclos e a necessidade de uma inicialização de $\mathbf{y}$.

Algoritmo 1.3.7 Multiplicação de matriz por vector (orientada por linhas)

```
proc MULTVR (A, x, m, n)
  comment: A ∈ R^{m×n}, R^m, y = Ax
  for i = 1 to m do
      s = 0                                   (acumulador)
      for j = 1 to n do
          s = s + a_ij x_j
      od
      y_i = s
  od
end
```

Algoritmo 1.3.8 Multiplicação de matriz por vector (orientada por colunas)

```
proc MULTVC (A, x, m, n)
  comment: A ∈ R^{m×n}, x ∈ R^n, y ∈ R^m, y = Ax
  for i = 1 to m do y_i = 0 od                (inicialização)
  for j = 1 to n do
      for i = 1 to m do
          y_i = y_i + a_ij x_j
      od
  od
end
```

Exercício 1.3.11 *Escreva um pseudo programa para calcular o produto* $\mathbf{A}^T\mathbf{x}$ *que evite formar explicitamente a transposta.*

Exercício 1.3.12 *Num problema surgem matrizes* $\mathbf{A} \in \mathbb{R}^{n\times n}$ *densas com valores de* n *tão grandes que tornam o armazenamento de* $\mathbf{A}$ *impossível. Sabe-se, no entanto, que* $a_{ij} = \alpha i + \beta j + \gamma$*, em que* $\alpha, \beta, \gamma \in \mathbb{R}$ *e são conhecidos e que o único tipo de operação que é necessário fazer é o produto de* $\mathbf{A}$ *por vectores* $\mathbf{x} \in \mathbb{R}^n$*. a) Deduza um método que torna trivial este cálculo; b) Escreva um pseudo programa que o implemente; c) Obtenha a respectiva complexidade temporal.*

O Algoritmo 1.3.9 traduz a operação de multiplicação de matrizes directamente a partir de definição. Tal como está escrito, a matriz $\mathbf{A}$ e $\mathbf{C}$ são acedidas por linhas e a matriz $\mathbf{B}$ por colunas, ou seja, o algoritmo efectua o produto interno da linha i de $\mathbf{A}$ pela coluna j de $\mathbf{B}$. Se o compilador armazena matrizes por colunas, então este algoritmo pode ser ineficiente do ponto de vista de paginação.

Exercício 1.3.13 *Considere o caso de um computador com páginas de* 1024 *Bytes e números reais ocupando palavras de* 8 *Bytes. Quantas multiplicações é que o Algoritmo*

1.3.7 executa antes de "saltar" de página? Estude o caso matrizes quadradas de ordem 10, 10^2 *e* 10^3.

Algoritmo 1.3.9 Multiplicação de matrizes (orientada por linhas)

```
proc MULTMR (A, B, C, m, n, p)
  comment: A ∈ ℝ^{m×p}, B ∈ ℝ^{p×n}, C ∈ ℝ^{m×n}, C = AB
  for i = 1 to m do
      for j = 1 to n do
          s = 0                                   (acumulador)
          for k = 1 to p
              s = s + a_ik b_kj
          od
          c_ij = s
      od
  od
end
```

Exercício 1.3.14 *Calcule a complexidade temporal, em flops, dos Algoritmos 1.3.7 e 1.3.9 em função de* m, p *e* n.

Uma pergunta legítima é a de saber se é possível modificar o Algoritmo 1.3.9 de modo a evitar paginação excessiva, ou seja, se é possível aceder às matrizes $\mathbf{A}$ e $\mathbf{C}$ por colunas. A resposta, baseada numa generalização directa do Algoritmo 1.3.8, é afirmativa e consta do Algoritmo 1.3.10.

Algoritmo 1.3.10 Multiplicação de matrizes (orientada por colunas)

```
proc MULTMC (A, B, C, m, n, p)
  comment: A ∈ ℝ^{m×p}, B ∈ ℝ^{p×n}, C ∈ ℝ^{m×n}, C = AB
  SMC (A, 0, m, n)                                (inicialização)
  for k = 1 to p
      for j = 1 to n do
          for i = 1 to m do
              c_ij = c_ij + a_ik b_kj
          od
      od
  od
end
```

Algoritmo de Strassen

Os métodos para multiplicar matrizes descritos atrás requerem todos $\mathcal{O}(n^3)$ flops. Uma pergunta legítima e oportuna é a seguinte: é o melhor que se pode fazer? Surpreendentemente, a resposta em STRASSEN (1969) é negativa. A ideia do algoritmo de Strassen constitui um caso típico da técnica de dividir para reinar.

Sejam $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n\times n}$ e $\mathbf{C} = \mathbf{AB}$ e n uma potência de 2 (adiante veremos como ultrapassar esta limitação). Particionemos cada uma destas matrizes em 4 submatrizes de ordem $n/2$, pondo

$$\mathbf{C} = \begin{pmatrix} \mathbf{C}_{11} & \mathbf{C}_{12} \\ \mathbf{C}_{21} & \mathbf{C}_{22} \end{pmatrix}, \quad \mathbf{A} = \begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} \mathbf{B}_{11} & \mathbf{B}_{12} \\ \mathbf{B}_{21} & \mathbf{B}_{22} \end{pmatrix} \tag{1.3.1}$$

Então, efectuando a multiplicação matricial, obtemos o conjunto de relações

$$\mathbf{C}_{ij} = \sum_{k=1}^{2} \mathbf{A}_{ik}\mathbf{B}_{kj} = \mathbf{A}_{i1}\mathbf{B}_{1j} + \mathbf{A}_{i2}\mathbf{B}_{2j}, \qquad i,j = 1,2 \tag{1.3.2}$$

Se designarmos por $F(n)$ o número de flops envolvido, nestas operações (8 multiplicações e 4 somas de matrizes de ordem $n/2$, deduzimos que

$$F(n) = 8F(n/2) + n^2$$

Não é difícil verificar, por simples substituição, que $F(n) = 2n^3 + \mathcal{O}(n^2)$, o que não revela nenhuma melhoria relativamente ao método convencional. Mas, em vez de desistir perante esta dificuldade, Strassen descobriu uma outra forma de organizar os cálculos em que produtos (caros) são trocados por somas e subtracções (mais baratos). Vejamos os detalhes. Pondo

$$\begin{aligned}
\mathbf{P}_1 &= (\mathbf{A}_{11} + \mathbf{A}_{22})(\mathbf{B}_{11} + \mathbf{B}_{22}) \\
\mathbf{P}_2 &= (\mathbf{A}_{21} + \mathbf{A}_{22})\mathbf{B}_{11} \\
\mathbf{P}_3 &= \mathbf{A}_{11}(\mathbf{B}_{12} - \mathbf{B}_{22}) \\
\mathbf{P}_4 &= \mathbf{A}_{22}(\mathbf{B}_{21} - \mathbf{B}_{11}) \\
\mathbf{P}_5 &= (\mathbf{A}_{11} + \mathbf{A}_{12})\mathbf{B}_{22} \\
\mathbf{P}_6 &= (\mathbf{A}_{21} - \mathbf{A}_{11})(\mathbf{B}_{11} + \mathbf{B}_{12}) \\
\mathbf{P}_7 &= (\mathbf{A}_{12} - \mathbf{A}_{22})(\mathbf{B}_{21} + \mathbf{B}_{22})
\end{aligned}$$

facilmente se conclui por substituição que

$$\begin{aligned}
\mathbf{C}_{11} &= \mathbf{P}_1 + \mathbf{P}_4 - \mathbf{P}_5 + \mathbf{P}_7 \\
\mathbf{C}_{12} &= \mathbf{P}_3 + \mathbf{P}_5 \\
\mathbf{C}_{21} &= \mathbf{P}_2 + \mathbf{P}_4 \\
\mathbf{C}_{22} &= \mathbf{P}_1 + \mathbf{P}_3 - \mathbf{P}_2 + \mathbf{P}_6
\end{aligned}$$

Aplicando este algoritmo recursivamente até chegarmos a matrizes de dimensão suficientemente pequena ($n = 2$, digamos), temos agora que o número de flops é determinado por

$$F(n) = 7F(n/2) + 9n^2 = 7F(n/2) + \mathcal{O}(n^2)$$

A solução desta relação de recorrência é

$$F(n) = cn^{\log_2 7} + \mathcal{O}(n^2) \approx \mathcal{O}(n^{2.81}) \tag{1.3.3}$$

uma melhoria relativamente a $\mathcal{O}(n^3)$. No entanto, verifica-se na prática que não só esta diferença não é suficiente para tornar este algoritmo competitivo com o método convencional como, por outro lado, o processo não permite tirar partido da eventual esparsidade das matrizes para economizar operações aritméticas. Daí que o algoritmo de Strassen tenha tido, até agora, apenas interesse teórico.

Exercício 1.3.15 *Determinar o número de flops do algoritmo de Strassen para o caso* $n = 2$.

Exercício 1.3.16 *Verificar a validade da expressão* (1.3.3).

2

Programação

2.1 Introdução

Um dos objectivos dos projectos de computação incluídos no Apêndice A é o de preparar profissionalmente os alunos para este género de trabalho. Assim, ao longo deste e do próximo capítulos, vão ser dadas algumas indicações visando:

- Promover a qualidade dos trabalhos efectuados;
- Contribuir para uma melhoria das capacidades dos alunos no tocante à realização de projectos e respectivos relatórios, as quais poderão ser úteis não só na escola e mas também na vida profissional;
- Estabelecer os critérios de aceitabilidade para os relatórios;
- Possibilitar aos alunos efectuar a auto-avaliação dos seus trabalhos, por forma a que as classificações por eles esperadas se aproximem das classificações atribuídas pelos docentes, bem como contribuir para a uniformização dos critérios aplicados por estes.

Estão aqui basicamente em jogo três tipos de competências: a compreensão de algoritmos, o desenvolvimento de programas e a escrita de relatórios. No presente capítulo concentrar-nos-emos no segundo tema, não no sentido de ensinar uma linguagem de programação, mas sim tendo em vista apresentar e explicar um conjunto de regras de boa prática.

2.2 Elaboração de programas

A programação é uma actividade intelectual complexa não obstante as falsas aparências que decorrem de exemplos demasiado didáticos em que uma escassa centena de instruções

chega para resolver o problema em mão. Na vida real, a dimensão dos programas mede-se em milhares e, mesmo, em milhões, de instruções: o *software* de controlo do vai-vém espacial americano ronda as 40 milhões de instruções; um sistema operativo moderno pode atingir 10 a 20 milhões de instruções![3]

É claro que projectos desta dimensão pressupõem equipas numerosas de programadores. Um dos aspectos mais controversos tem a ver com o tempo que estes projectos levam a completar. As estimativas, de modo geral, pecam por defeito, i.e., os programas demoram sempre muito mais tempo a ficar prontos do que foi planeado. Nos grandes projectos, empregando equipas de programadores, nem sempre o aumento do número destes se traduz numa redução proporcional do prazo já que o trabalho de coordenação da equipa também cresce com a respectiva dimensão. Uma regra prática – que vale o que vale – diz que, se um projecto leva t_1 unidades de tempo a ser feito por um programador, com n programadores levará $t_n = 2t_1/(n+1)$. Isto quer dizer que o reforço de uma equipa de dois programadores com um terceiro apenas permitirá ganhar 25% de tempo e não 33% como poderia parecer na base de uma simples proporcionalidade.[4]

Exercício 2.2.1 *Comparar, em termos de tempo e de custo, as seguintes duas hipóteses para um projecto de programação:*

- *uma equipa de 10 programadores;*
- *divisão do projecto em dois sub-projectos distintos, cada um com uma equipa de 5 programadores trabalhando em paralelo.*

Explicar a frase (dita em tom lamuriento): "um projecto grande acaba por custar sempre o dobro do que custaria se fosse feito por um único programador".

Exercício 2.2.2

1. *Qual a diferença entre algoritmo e programa?*
2. *O que é uma estrutura de dados e o que é que a distingue das variáveis matemáticas? Dê exemplos.*

Podem identificar-se quatro fases essenciais no desenvolvimento de um programa:

1. Análise – traduzir o algoritmo matemático num programa e especificar as funcionalidades deste, ou seja, definir precisamente o que o programa deve ser capaz de fazer.
2. Organização e estrutura dos dados – associar as variáveis matemáticas do problema às variáveis a definir no programa.
3. Codificação – a escrita do programa na linguagem e para o ambiente computacional escolhidos.
4. Teste ou validação – a verificação de que o programa executa, correcta e eficazmente, as tarefas especificadas.

Estes diferentes aspectos vão ser desenvolvidos a seguir.

2.2.1 Atributos de um bom programa

Os atributos fundamentais de um programa para resolver um problema de cálculo numérico (e não só) são os seguintes:

Correcção

A correcção é a qualidade que garante que o programa resolve correctamente o problema proposto, i.e., que produz o resultado certo para cada instância possível dos dados de entrada. Por isso constitui o atributo mais importante já que nenhuma outra qualidade será relevante se o programa não "funcionar". A correcção obtém-se recorrendo a diversas técnicas:

- Programação estruturada para tornar evidente a lógica do programa;
- Realização de testes de validação que executem todos os segmentos de código. Por exemplo, nos blocos do tipo $IF-THEN-ELSE$ todas as opções possíveis devem ser testadas para se poder concluir que o programa está correcto.

Clareza

Um programa é a tradução, numa linguagem apropriada, de um algoritmo. Daí que antes de começar a programar há que compreender cabalmente o algoritmo e o seu funcionamento. Para além de ser um conjunto sequencial de instruções destinadas a ser executadas por uma máquina, um programa é também um instrumento de comunicação de ideias. Neste sentido, um programa – que se destina a ser executado por uma máquina – deve poder ser lido e entendido também por seres humanos. Por isso, deve usar-se um estilo de programação que torne evidente o que o programa faz e como o faz. Um programa bem escrito deve permitir a sua leitura sem esforço, de modo a poder ser facilmente compreendido, utilizado, mantido e modificado. Eis algumas sugestões para a construção de programas fáceis de perceber:

- Identificar, no início de cada módulo (programa principal, subrotina ou função), a tarefa executada e as variáveis de entrada e de saída;
- Assegurar uma correspondência clara entre o algoritmo e o programa que o implementa;
- Usar nomes mnemónicos para as variáveis. Por exemplo, o multiplicador utilizado para condensar uma linha no método de Gauss pode ser designado por $MULT$ ou MIK mas devem evitar-se designações como $TEMP3$ (não é um nome mnemónico) ou $MULTIPLICADORLINHAK$ (mnemónico mas muito extenso);

- Não recorrer a demasiados níveis de parênteses (cinco níveis parece ser o máximo que a mente humana consegue apreender sem ficar confusa).
  Exemplo (mau):

$$y = b(2 + (5(a + (\sin(1/(c(1 + x^2))))))/c)$$

  Exemplo (melhor, partindo a expressão em duas):

$$z = \sin(1/(c(1 + x^2)))$$
$$y = b(2 + 5(a + z)/c)$$

- Evitar interfaces com o utilizador demasiado elaboradas, com mais linhas de código do que as próprias subrotinas que efectuam os cálculos.

Simplicidade

A simplicidade significa que o programa não deve ser mais complicado que o necessário ao cumprimento dos objectivos especificados. É um erro comum dos programadores novatos querer mostrar competência com truques que só marginalmente aumentam a eficiência (quando aumentam!) e, de certeza, pioram a legibilidade do programa já para não falar da robustez. Não esquecer nunca as máximas:[5]

KIS – *Keep It Simple!*

ou, melhor ainda,

KISS – *Keep It Simple Stupid!*

Modularidade

O princípio orientador da *modularidade* ou segmentação é o de fazer corresponder a cada tarefa logicamente distinta um módulo ou subprograma. Vem a propósito, invocar outra máxima de programação, adaptada de uma campanha presidencial americana em que se insinuava que um certo candidato era tão estúpido que não conseguia mascar pastilha elástica e subir escadas ao mesmo tempo:

Do not chew gum while climbing stairs!

ou seja, fazer uma tarefa de cada vez, i.e., uma tarefa em cada subprograma.

Todavia, deve-se resistir a impulsos extremos: uma segmentação demasiada fina (muitos subprogramas mas cada um executando uma tarefa trivial) ou "meter tudo" no programa principal.

Robustez

Por robustez entende-se a capacidade do programa para resistir a "maus tratos" infligidos por utilizadores que não estão familiarizados com a respectiva utilização. Um programa robusto deve assinalar, por meio de mensagens adequadas, as anomalias detectadas durante a sua execução. Por exemplo, numa subrotina para solução de equações não lineares, deverão ser assinaladas de forma apropriada e diferenciada tentativas de divisão por zero, ultrapassagem do número máximo de iterações, situações de divergência, etc.

Eficiência

A eficiência consiste em efectuar as tarefas necessárias com economia de recursos computacionais (tempo, memória). *Exemplos*: calcular o valor de um polinómio para um dado valor do argumento pelo algoritmo de Horner, evitar calcular a inversa de uma matriz, reduzir acessos a memórias periféricas lentas (discos), etc. Deve ter-se em atenção que tipo de recurso se quer privilegiar (tempo de cálculo, memória ou outro?) e que, por vezes, é preciso aceitar compromissos uma vez que não é possível ter tudo.

2.2.2 Estilo de programação

Para que os programas produzidos possuam um mínimo de qualidade é conveniente observar os seguintes preceitos:

- Em todos os módulos de programação (funções, subrotinas, etc), devem existir:
  - Uma *secção de interface* onde conste a informação necessária para se poder utilizar o módulo. Esta secção deve conter:
    - a identificação do(s) programador(es), versão, data, etc.
    - a descrição da função ou tarefa executada pelo módulo e a lista de subrotinas externas que sejam eventualmente referenciadas;
    - as declarações das variáveis de *input* e *output* (variáveis globais) e das variáveis que são utilizadas internamente (variáveis locais);
    - a respectiva descrição, por meio de comentários, e inicialização, se for caso disso.
  - Uma *secção de implementação* onde se inclui o código propriamente dito com as instruções que implementam a tarefa correspondente à subrotina, bem como comentários que auxiliem a compreensão.
- Não devem ser utilizados estilos de programação tortuosos, que, embora possam abonar em favor da habilidade do programador, resultem em código confuso ou críptico. Sempre que se revele altamente vantajoso usar truques, esse facto deve ser assinalado por meio de comentários esclarecedores;

- É boa prática declarar explicitamente todas as variáveis por meio de instruções apropriadas (REAL, INTEGER, ..., em FORTRAN; double, int, ..., em C; etc.), mesmo que a linguagem utilizada possa dispensar este procedimento;

- Não devem ser utilizadas constantes no seio do programa. Os valores constantes estritamente indispensáveis devem ser substituídos por variáveis inicializadas na secção apropriada do código ou por parâmetros;

- Para aumentar a legibilidade do código, deve ser usada indentação nos ciclos e blocos de modo a tornar evidente a respectiva estrutura lógica;

- Sempre que sejam declarados *arrays* como variáveis locais num subprograma, devem ser indicadas as eventuais limitações relativamente às suas dimensões. Os *arrays* usados como espaço de trabalho devem fazer parte da interface do subprograma (sendo portanto incluídos no conjunto da variáveis globais). Este procedimento transfere para o utilizador a responsabilidade pela gestão correcta do espaço de memória utilizado;

- Devem ser utilizados subprogramas para as operações elementares mais frequentes; por exemplo, as subrotinas da programateca BLAS (ver ASHBY et al. (1990)) para as operações básicas da Álgebra Linear). Este procedimento facilita a construção modular e em geral permite aumentar a legibilidade e a eficiência.

Exercício 2.2.3 *Para guardar o valor da constante π, um programador criou a variável PI e inicializou-a a $PI = 3.1416$; outro programador decidiu-se por fazer $PI = 4\arctan(1.0)$. Comente estas duas opções explicando as respectivas vantagens e desvantagens.*

Identificação do programa

Um programa deve sempre conter, à cabeça, um bloco com a seguinte informação:

- a identidade dos autores;

- a data;

- uma descrição sumária do problema que é suposto resolver e dos métodos usados;

- requisitos e limitações.

Isto parece uma duplicação do que se pede no relatório (ver a Secção 3.2). No entanto, convém não esquecer que o programa e o relatório podem seguir caminhos diferentes, i.e, ser lidos por pessoas diferentes em alturas diferentes. Eis um exemplo:

Autores:	Ana B. Silva e Pedro C. Gomes
Data:	2001.10.24
Descrição:	Integração adaptativa iterativa de uma função f contínua num intervalo finito $[a, b]$ usando a regra do trapézio (ver PINA (1995)).
Requisitos:	A função f deve ser dada por uma *FUNCTION FUN(X)* escrita pelo utilizador.

Divertimento

Aqui se deixam alguns conselhos para quem pretenda escrever programas verdadeiramente ilegíveis, impossíveis de afinar e de manter.

- Minta nos comentários: estes podem dizer uma coisa e o código fazer outra. Muitas vezes isto consegue-se sem esforço adicional: basta ir alterando o código durante o seu desenvolvimento e não ajustar concomitantemente os comentários.

- Inclua comentários indispensáveis como "somar 1 a I", mas evite dizer para que serve uma subrotina ou uma função: nada de facilitar a vida a estranhos (o próprio programador é um estranho um mês após ter escrito o programa!).

- Sempre que escolher SMN para nome de variável não esquecer de usar também SNM como nome de (outra) variável; mas melhor ainda são $I0$ e IO, $l1$ e Ll, $DOCalc$ e $D0CalC$.

- Se o algoritmo usar a letra h para denotar o passo, este deve ser designado por $J3y$ no programa – assim, ninguém vai adivinhar e os agentes das potências inimigas hão-de passar um mau bocado até descobrir o truque.

- O papel deve ser bem aproveitado: espaços em branco, linhas curtas e indentação facilitam a leitura, o que os verdadeiros profissionais, não sem uma ponta de orgulho, dispensam.

- Se for preciso recorrer frequentemente ao valor de π não defina uma variável PI que contenha este valor; deste modo, pode usar diferentes aproximações 3.14, 3.1416, etc., em diferentes partes do programa e assim evitar a monotonia que aflige muitos textos.

- Nunca recorrer aos humildes símbolos I, J e K para variáveis de controlo de ciclos; use de preferência $P34T$, $UYT5$ ou $ERDF$.

- Uma subrotina ou procedimento deve sempre produzir efeitos colaterais; assim, se escrever uma subrotina para calcular a norma euclideana de um vector não se esqueça de a aproveitar para apagar o ficheiro de dados.

- Se o programa foi escrito para resolver sistemas de equações lineares com 20 equações, foi *mesmo* escrito para 20 equações: se alguém tiver o desplante de querer resolver sistemas com 10 equações, então que modifique o programa já que são só precisas 47 alterações, estrategicamente camufladas e não documentadas (era o que faltava!).

Mas há mais (apanhados na Internet):

- DEBUGGING: Removing the needles from the haystack;
- Endless Loop: n., see Loop, Endless. Loop, Endless: n., see Endless Loop.
- Any sufficiently advanced bug is indistinguishable from a feature.
- The computer is mightier than the pen, the sword, and usually the programmer.
- After a number of decimal places, who cares?
- There are two ways to write error-free programs; only the third one works.
- You never finish a program, you just stop working on it.
- Deliver yesterday, code today, think tomorrow.

2.2.3 Validação dos programas

Um aspecto da maior importância consiste na *validação* dos programas, i.e., em assegurar que executam correctamente as tarefas para as quais foram desenvolvidos. A validação deve ser feita sequencialmente em várias fases:

1. Afinação da correcção sintática do programa. Um bom compilador produz mensagens e avisos geralmente suficientes para detectar e corrigir este tipo de erros.

2. Verificação, por meio de testes, que a lógica do programa está correcta, i.e., conseguir que o programa produza os primeiros resultados certos.

   Nesta fase deve começar-se por casos de teste simples, mesmo triviais. Por exemplo, se o programa for destinado a construir polinómios interpoladores de grau n qualquer, deve começar-se por testar os casos $n = 0$ e $n = 1$; num programa para resolver equações não-lineares não faz sentido testar a equação $\exp(1+\sin x^2)+\ln(1+x^4)-5 = 0$ antes de saber se o programa consegue resolver a equação $2x-1 = 0$; se o programa se destinar a resolver sistemas de equações lineares, deve testar-se primeiro os casos de uma e duas equações. Só quando se conseguir ultrapassar esta barreira é que vale a pena prosseguir para a fase seguinte.

3. Uma vez passados estes testes simples, pode então avançar-se para os casos mais complicados, para os casos especiais ou, mesmo, anómalos, e, finalmente para os casos pedidos no enunciado e outros considerados relevantes. É importante empregar uma bateria de testes que percorra todas as funcionalidades do programa.

4. Por fim, deve proceder-se ao melhoramento ou optimização do programa: torná-lo mais rápido, menos consumidor de memória, etc., consoante o objectivo visado.

3

Relatórios

3.1 Introdução

Neste capítulo são apresentadas a estrutura geral e as normas a que os relatórios de projectos computacionais devem obedecer. Muito do que aqui vai ficar dito aplica-se, com as devidas adapatações, a qualquer relatório técnico, já que estes textos tendem a assumir uma estrutura mais ou menos comum e consagrada.

> *Reading maketh a full man; conference a ready man; and writing an exact man.*
>
> – Francis Bacon (1561-1626)

3.2 Escrita de relatórios

O objectivo desta secção é o de apresentar algumas ideias e conselhos úteis na elaboração de relatórios de natureza técnica ou científica.[6]

Devemos ter consciência que é sobretudo através do relatório que o projecto é avaliado: um mau relatório pode retirar valor a um bom trabalho. Como peça de comunicação escrita, deve ser redigido tomando sempre os seguintes cuidados:

- estilo de escrita claro, directo, conciso e convincente;
- gramática, ortografia e pontuação correctas;
- apresentação gráfica simples mas eficaz.

Quanto ao primeiro aspecto, só há uma receita para melhorar o nosso estilo pessoal de escrita: ler regularmente prosa de bons autores. Quanto aos dois últimos aspectos, com as ferramentas hoje disponíveis em qualquer computador pessoal (editores e processadores de texto, correctores ortográficos, etc.), não são desculpáveis quaisquer falhas.

3.2.1 Formato e estrutura

A seguinte formatação geral para o relatório dos projectos computacionais é adequada:

- Texto em páginas de formato A4 e numeradas (excepto a página de rosto);
- Fonte romana de 12 pontos (podendo ser de 10 ou 11 pontos localmente);
- Espaçamento simples entre linhas;
- Margem esquerda com cerca de 4 cm e margens direita, superior e inferior com cerca de 2.5 cm;
- Capítulos, secções, figuras e tabelas numerados;
- Capa que não modifique as dimensões do documento base; por exemplo, capa e contra-capa em plástico transparente (de modo a permitir que a página de rosto seja visível), presas com argolas ou coladas;
- Evitar folhas soltas ou diskettes que se destaquem facilmente do documento base (para incluir diskettes usar bolsas apropriadas disponíveis no mercado).

Recomenda-se também a adopção da seguinte estrutura:

- Página de rosto, de acordo com o modelo da Secção 3.2.1;
- Índice paginado, na primeira página do relatório;
- Resumo;
- Introdução;
- Formulação teórica;
- Descrição do algoritmo e do programa;
- Resultados obtidos;
- Discussão;
- Conclusões;
- Agradecimentos;
- Referências bibliográficas;
- Anexos.

Vejamos em pormenor os vários aspectos de um relatório.

3.2.2 Resumo

O Resumo é uma componente vital de qualquer documento técnico ou científico, uma vez que a maior parte das vezes (especialmente em revistas técnicas e actas de conferências) a sua leitura poderá decidir se o resto do texto merece ser lido ou não. Acresce ainda que, em certas circunstâncias, somente o Resumo é publicado, pelo que a sua qualidade é determinante do êxito ou fracasso da divulgação do trabalho efectuado.

Pode considerar-se que os objectivos do Resumo são idênticos aos da Introdução, pelo que, de uma forma geral, os mesmos princípios são aplicáveis. No entanto, aqui entra também em jogo um novo elemento: o constrangimento imposto pelo factor espaço. Dado que o Resumo é, por natureza, muito mais curto do que a Introdução, o principal problema é o da selecção dos aspectos a referir. Assim, a clareza e a concisão na redacção são os principais atributos a visar na sua composição. O Resumo não deve conter definições e fórmulas, a menos que sejam totalmente imprescindíveis, e nunca deve incluir aspectos que não tenham sido essenciais para a realização do trabalho ou que não se encontrem desenvolvidos em outras secções do relatório. Um Resumo de um trabalho desta natureza deve ter 10 a 15 linhas, no máximo 20, pelo que cada palavra conta.

3.2.3 Introdução

Uma boa Introdução serve para suportar todo o restante conteúdo do relatório e, por isso mesmo, é, tal como o Resumo, quase sempre uma das últimas secções a escrever.

Para redigir uma boa Introdução, o autor deve colocar-se na posição do leitor, tendo em mente que os objectivos da respectiva leitura são os seguintes:

- compreender de forma clara e completa o problema tratado;
- ficar a conhecer, em linhas gerais, como foi abordado e resolvido o problema;
- avaliar a aplicabilidade, a originalidade e a generalidade do trabalho;
- reconhecer a organização do relatório, com vista a uma eventual leitura selectiva;
- apreender, ainda que de forma superficial, a essência dos resultados obtidos (sem contudo cair em redundância relativamente à Discussão dos resultados e às Conclusões).

Assim, numa Introdução, é correcto incluir informação relativa a:

- enquadramento do problema a resolver na matéria leccionada na disciplina;
- princípios e orientações gerais seguidos no estudo do problema e na construção do programa;

- aspectos mais interessantes ou originais, ou que tenham aplicação na solução de outros problemas;
- resultados mais importantes obtidos no estudo;
- estrutura do relatório (para permitir uma leitura selectiva).

Numa Introdução deve-se evitar:

- confundir Introdução com Formulação teórica;
- consumir espaço com generalidades, mais próprias num livro de texto, ou com transcrições mais ou menos trabalhadas desses mesmos livros ou de apontamentos das aulas;
- incluir matéria ou assuntos que não tiveram relação directa com o trabalho propriamente dito ou com o problema proposto, só para "encher".

3.2.4 Formulação teórica

O objectivo da secção Formulação teórica é descrever a teoria que suporta a resolução do problema proposto, sendo o seu conteúdo muito variável. Por exemplo, num problema de interpolação é correcto incluir:

- uma descrição breve do problema geral da interpolação e do âmbito de aplicação desta técnica;
- uma descrição, também sucinta, dos aspectos gerais da teoria da construção de polinómios interpoladores;
- uma explicação das ligações entre os vários aspectos da teoria apresentada;
- uma descrição sumária dos resultados disponíveis para cálculo de estimadores para o erro de interpolação;
- uma menção às vantagens e limitações da técnica utilizada.

Nesta secção devem ser introduzidos os conceitos fundamentais, as definições e as equações relevantes. É essencial que todos os símbolos, variáveis, etc., que figuram nas equações sejam definidos, para que não haja dúvidas de interpretação.

É necessário resistir à tentação de efectuar transcrições teóricas extensas, convindo antes dar especial ênfase a aspectos que não se encontram tratados nos livros de texto mas que tenham sido desenvolvidos para a resolução do problema proposto. Há que ter em mente que um relatório técnico e um livro de texto visam objectivos diferentes.

O texto da secção Formulação teórica deve assegurar de forma perfeita a continuidade na transição para a secção seguinte (Descrição do algoritmo e do programa) marcando, no entanto, as diferenças entre estas duas fases distintas do trabalho: a teoria de suporte e a implementação computacional.

3.2.5 Descrição do algoritmo e do programa

Esta secção deve ser redigida tendo como objectivo principal explicar o método de cálculo utilizado, de modo a permitir uma compreensão e leitura fácil do programa que implementa o método, fazendo portanto parte da documentação externa do programa.

Tipicamente, o código construído para resolver um problema prático compreende um programa principal e diversas subrotinas que executam tarefas específicas. Assim, esta secção deve conter primeiramente uma descrição geral da estrutura do programa principal e das subrotinas, completada por uma ou mais figuras contendo um esquema (ou fluxograma) que ilustre a inter-relação entre o programa principal e as diversas subrotinas. Deve haver o máximo cuidado na elaboração deste esquema para que ele seja efectivamente informativo e sugestivo e não meramente decorativo. A seguir deve vir a descrição sucinta de cada uma das subrotinas com a seguinte informação:

- função executada pela subrotina;
- o formato de invocação (ou instrução de chamada da subrotina), o qual contém o nome do subprograma e todos os argumentos de entrada e saída (*Exemplo*: $CALL\ LUFACT\,(A, NMAX, N, PIVOT, IERR)$, na linguagem FORTRAN);
- uma descrição de todos os argumentos (variáveis de *input* e de *output*) incluídos na instrução de chamada da subrotina, a qual compreende o seu nome, tipo e significado (que informação armazena, se a variável é de *input* ou de output, como é modificada na subrotina, etc.);
- uma descrição das variáveis definidas internamente, se a importância dessas variáveis no contexto do algoritmo assim o justificar;
- uma indicação do encadeamento com outras subrotinas, se for caso disso (por exemplo, $LUSOLV$ é chamada após a chamada a $LUFACT$ se $IERR = O$);
- uma descrição da sequência de cálculos efectuada, i.e., do algoritmo codificado pela subrotina, descrito por palavras e repetindo as fórmulas pertinentes (as quais já terão sido introduzidas na secção Formulação teórica). Para facilitar a compreensão e leitura do programa poderão ser apresentados esquemas ou fluxogramas das subrotinas consideradas mais importantes;
- notas que sejam importantes para a utilização da subrotina (por *Exemplo*, se numa subrotina de cálculo dos momentos de um *spline* é declarado internamente um vector que limita o número máximo de nós utilizável, esse facto deve ser referido).

3.2.6 Resultados

As secções Resultados e Discussão têm obviamente um peso decisivo em qualquer trabalho científico ou técnico. São aspectos fundamentais a selecção dos resultados a incluir e a sua forma de apresentação. A componente gráfica da apresentação ("uma imagem vale por mil palavras!") é uma arte mas muito pode ser aprendido e muitos erros evitados.[7]

Na secção de Resultados deve-se começar por referir quais os casos de estudo usados nos testes e na fase de exploração do(s) programa(s) e só então se passar aos resultados obtidos nos Exemplos do enunciado.

Devem ser tidos em conta alguns aspectos, nomeadamente:

- é necessário seleccionar criteriosamente os resultados que se devem incluir no relatório. Se forem incluídos (e devem ser!) resultados para além dos estritamente pedidos no enunciado, deve ser explicada a estratégia de exploração seguida, i.e., o motivo pelo qual tais resultados foram incluídos no relatório;

- é inútil incluir longas listas de números, pois estas não serão lidas nem permitem a quem lê o trabalho extrair as suas próprias conclusões;

- sempre que possível, as listas devem ser substituídas por tabelas ou figuras, cuja interpretação seja mais fácil;

- deve evitar-se a apresentação de elementos redundantes. Por exemplo, se se incluir uma figura com os erros de interpolação, então não se deve incluir também uma tabela contendo a mesma informação;

- não se deve apresentar como figura ou tabela um resultado que possa ser condensado numa frase simples ou numa expressão analítica.

Para todos os casos estudados – os pedidos no enunciado e outros que também tenham sido analisados – devem ser apresentados:

- os dados de entrada para permitir a reprodução dos resultados. Esta descrição pode ser complementada com uma figura ou anexo contendo uma cópia dos ficheiros de *input*;

- os resultados, sob a forma de números, *outputs* do programa, tabelas e figuras, etc;

- anotações e comentários referindo os aspectos mais significativos dos resultados, se não forem evidentes a partir da leitura dos *outputs*, tabelas ou figuras, mas sem cair em redundância relativamente à secção que normalmente se segue (Discussão).

3.2.7 Discussão

A secção Discussão desempenha num relatório técnico um papel fundamental por duas razões: por um lado, é nesta secção que o autor faz a interpretação dos resultados obtidos; por outro lado, é sempre uma espécie de ponte entre os resultados e as conclusões.

A Discussão propriamente dita é muitas vezes fundida com a apresentação dos resultados, sem que tal prejudique a qualidade e clareza do relatório. No entanto, convém ter presente que a discussão dos resultados tem que existir sempre, e não pode nunca ser uma recapitulação mais ou menos óbvia destes.

Numa Discussão interpreta-se, não se recapitula. É vital nesta secção responder às perguntas colocadas no enunciado ou suscitadas no decurso do trabalho. Numa Discussão deve-se explicar em termos globais em que medida os resultados coincidem com ou se desviam dos resultados esperados ou conhecidos da literatura, i.e., deve ser desenvolvida uma análise comparativa e qualitativa. Uma situação de desvio relativamente ao esperado teoricamente pode não indiciar um erro de programação (que deveria ter sido detectado e eliminado nos testes) mas sim um problema com particularidades interessantes, a merecer aprofundamento.

Na redacção desta secção, devem evitar-se expressões vagas e pouco rigorosas. Expressões tais como "O erro foi geralmente muito pequeno" não são adequadas. Uma expressão correcta seria, por exemplo, "em todos os casos de teste, o erro foi inferior a 0.1%". Para além de ser completamente objectiva ("muito pequeno" depende do contexto, 0.1% não), esta expressão mostra que o autor compilou e analisou devidamente os valores numéricos obtidos, que é precisamente o que se pretende.

Na Discussão é apropriado incluir figuras e tabelas que, construídas a partir dos resultados, apresentem novos pontos de vista, contenham informação nova ou dificilmente dedutível e que, portanto, auxiliem o leitor na interpretação dos resultados. Raramente se devem incluir listagens na Discussão.

Ao redigir esta secção é necessário ter o máximo cuidado para não introduzir interpretações ou afirmações duvidosas que não possam ser efectivamente deduzidas dos resultados. A invenção e a especulação não têm lugar em relatórios técnicos.

Também deve evitar-se introduzir, ainda que de forma implícita, juízos de valor sobre os resultados obtidos ou sobre o mérito do trabalho. *Exemplo* (mau): "Os resultados obtidos foram de grande qualidade, tendo sido plenamente atingidos todos os objectivos do trabalho". A qualidade do trabalho é, em última análise, julgada pelos leitores, não pelo autor.

3.2.8 Conclusões

A secção Conclusões constitui o culminar do relatório e é, por isso, de uma importância primordial. Nesta secção devem ser recapitulados todos os aspectos do trabalho, com ênfase especial na discussão dos resultados. É importante ser sintético, por vezes quase

tão conciso como no Resumo.

Deve existir o máximo cuidado para não inventar conclusões. Resultados bons, correctos, bem seleccionados e discutidos, proporcionam geralmente conclusões rigorosas e concisas. É este objectivo que se deve procurar atingir num trabalho desta natureza, não sendo aceitáveis conclusões "caídas do céu em pára-quedas".

Após redigir esta secção, é necessário verificar se ela inclui as respostas a todas as questões levantadas pelo enunciado, pois em caso contrário haverá que reformular o texto (ou o trabalho!). Devem também ser apresentadas respostas para as questões que o autor achou de interesse no decurso do trabalho.

Nas Conclusões é apropriado referir quais os aspectos do problema proposto que não foram explorados, quais os objectivos que não foram atingidos e quais as questões levantadas pelos resultados que não tiveram resposta adequada. Por outras palavras, deve-se propor quais os estudos de continuação que, na opinião do autor, deveriam ser conduzidos posteriormente com base no trabalho efectuado. Por isso, a secção Conclusões surge por vezes referida como "Conclusões e desenvolvimentos futuros".

3.2.9 Agradecimentos

Para a elaboração de um trabalho contribuem naturalmente os seus autores mas eventualmente também outras pessoas ou instituições. Faz parte da ética mencionar e agradecer estas contribuições de terceiros, o que deve ser feito de forma clara e sóbria.

Exemplo (bom): "Agradecemos ao Centro de Informática do IST pelo acesso ao seu supercomputador OMEGA e à nossa colega Ana B. Silva da Universidade do Sul pela disponibilização do seu programa GRAFIX com o qual foram elaborados os gráficos. O presente trabalho foi desenvolvido no âmbito do Projecto ABC123/2000 da European Foundation for Young Researchers".

Exemplo (mau): "Agradecemos ao nosso Prof. Augusto B. Silva a sua inescedível paciência sem o que não teria sido possível concluir este trabalho em tempo útil e com qualidade".

Exemplo (muito melhor): "Agradecemos ao Prof. Augusto B. Silva do Departamento de Matemática do IST pela ajuda na implementação do método de More-Swift em computadores massivamente paralelos e ao Dr. Carlos D. Schwab do LDMEC pela tradução para português da referência SCHNAPS97".

3.2.10 Referências bibliográficas

A apresentação de uma lista de referências bibliográficas nos relatórios é obrigatória pois não é eticamente correcto omitir os elementos a que se recorreu para elaborar o trabalho.

Existem muitas normas para a forma de apresentação de referências, no texto e na lista bibliográfica, não sendo possível nem desejável um modelo rígido. É importante,

contudo, manter a uniformidade e a coerência ao longo de todo o relatório e na lista de referências. Assim, sugerem-se as seguintes formas muito usuais:

[18] A. Uthor, A new approach to matrix inversion, J. of Lin. Algebra, 28, 123-132, 1956.

[FERREIRA87] J. Campos Ferreira, Introdução à Análise Matemática, Gulbenkian, 1987.

as quais devem ser citadas no texto como [18] e [FERREIRA87], respectivamente. Convém notar que a primeira, [18], obriga a consultar a lista de referências para descobrir o significado do '18', o que pode constituir um arreliador desvio de atenção; a segunda, [FERREIRA87], poupa este esforço ao leitor. A variante utilizada na Bibliografia do presente texto também é muito frequente.

3.2.11 Anexos

Deve ser incluído como Anexo tudo o que tenha importância suficiente para figurar no relatório mas que pela sua especificidade ou volume é susceptível de "cortar o fio à meada" durante a leitura. Pela mesma ordem de ideias, nunca deve ser incluída como Anexo informação própria da organização de um relatório técnico (por exemplo, resultados, descrição do algoritmo, etc.).

Os Anexos devem ser designados por Anexo A, Anexo B, etc., podendo ser ordenados pela sequência com que são referidos no corpo do relatório.

Nos relatórios dos projectos computacionais, é apropriado incluir os seguintes Anexos:

- Demonstrações de teoremas, se forem necessárias.

  Relativamente às demonstrações, deve procurar-se a maior concisão possível, desde que a clareza não seja prejudicada. Se houver necessidade de introduzir demonstrações de tópicos diferentes (por exemplo, Cálculo e Álgebra Linear) devem utilizar-se Anexos separados.

- Listagens dos programas.

  Caso haja lugar a alterações no código fonte consoante os problemas estudados (a evitar, em princípio), devem ser incluídas listagens das instruções alteradas, indicando rigorosamente o ponto do código no qual as alterações foram introduzidas.

- Listagem de ficheiros de *input* e de *output*;

  As listagens dos ficheiros de *input* destinam-se a proporcionar ao leitor informação factual e completa de todos os parâmetros usados nos casos de teste do programa. Estas listagens devem ser equilibradas, i.e., não devem ser incluídas listagens longas que não possam ser lidas em tempo útil.

- "Manual do utilizador" do programa e instruções para instalação e operação, se for caso disso.

  A inclusão do "Manual do Utilizador" do programa não é obrigatória e a sua importância no âmbito dos objectivos desta disciplina é secundária relativamente à descrição do algoritmo e do programa.

- Diskette contendo o código fonte, código executável e ficheiros de *input* e de *output*.

  É conveniente criar directorias separadas para o código fonte, código executável, ficheiros de *input* e ficheiros de *output*. A diskette deve ser guardada numa bolsa apropriada de modo a que não se destaque facilmente do resto do documento.

3.2.12 Revisão

Um relatório não deve ser dado por concluído sem que se tenha procedido à sua leitura crítica e à sua verificação. A experiência mostra que muitos erros, falhas e imprecisões podem ser eliminados numa leitura final cuidada. Os aspectos essenciais a ter em conta são:

1. Reler o texto como se não tivesse sido o seu autor e, em especial, verificar se contém as respostas às questões concretas colocadas no enunciado;

2. Verificar o número de páginas do relatório e das suas componentes. Os relatórios dos projectos computacionais devem ter em média 15 a 20 páginas, excluindo os Anexos, e 20 a 30 páginas no total, limites que não devem ser interpretados como uma imposição. No entanto, se o relatório tiver 5 ou 100 páginas deve ser reformulado;

3. Verificar se o índice geral e os conteúdos das secções correspondem aos recomendados e, em particular, se as diversas secções estão na ordem correcta;

4. Os programas devem ter listagens de 2 a 5 páginas. Se tal não acontecer, talvez seja sinal de que as recomendações não foram seguidas;

5. Rever as referências bibliográficas de modo a garantir que estas obedecem a um estilo uniforme e que todas se encontram citadas algures no relatório;

6. Percorrer todos os pontos da seguinte lista de verificação:

   (a) Aspecto correcto (formato, fontes, etc.);

   (b) Página de rosto, segundo o modelo da Figura 3.2.1;

   (c) Índice paginado;

   (d) Listas de figuras e de tabelas;

(e) Conteúdo: Resumo, Introdução, Descrição do algoritmo e do programa, Resultados, Discussão, Conclusões, Agradecimentos, Referências, Anexos (Listagens de programas, etc.);

(f) Programa legível e documentado;

(g) Número de páginas equilibrado;

(h) Não há peças (folhas ou diskettes) soltas.

INSTITUTO SUPERIOR TÉCNICO

Métodos Computacionais

Licenciatura em: Ano lectivo: Turma:

- Aluno (número e nome):
- Assinatura:
- Aluno (número e nome):
- Assinatura:

Projecto X.1.2

Cálculo de *splines* cúbicos

1) Desenvolva um programa para o cálculo de *splines* cúbicos com nós equidistantes. Aplique este programa a alguns casos de teste, nomeadamente à interpolação das seguintes funções:

a) $f(x) = \cos x, \quad \bar{\Omega} = [0, \pi/2]$

b) $f(x) = (1 - x^2)^{1/2}, \quad \bar{\Omega} = [0, \sqrt{3}/2]$

c) $f(x) = x^4, \quad \bar{\Omega} = [0, 2]$

2) Determine o erro cometido na aproximação à função f e às suas derivadas f' e f''. Mostre a sua evolução com o espaçamento dos nós. (Tome para y'_0 e y'_n valores apropriados às funções interpoladas.)
3) Estime a ordem de convergência e compare-a com o valor teórico.

Figura 3.2.1: Página de rosto

3.3 Classificação dos projectos computacionais

Nesta secção é descrita a fórmula de cálculo utilizada para obter a nota final dos projectos computacionais. Com este procedimento, pretende-se, por um lado, uniformizar e tornar mais objectiva a avaliação dos projectos computacionais por parte dos docentes da disciplina e, por outro lado, proporcionar aos alunos um método de auto-avaliação.

A classificação dos projectos computacionais é obtida através da classificação, na escala 0-5, dos seguintes aspectos:

Relatório:

- Organização e apresentação do relatório como peça escrita, R_O;
- Descrição do algoritmo e do programa, R_A;
- Resultados e exploração do programa, R_R;
- Discussão, R_D.

Programa:

- Correcção, P_C;
- Legibilidade, P_L;
- Robustez, P_R;
- Eficiência, P_E.

A nota C do projecto computacional, na escala 0-20, resulta da seguinte média ponderada:

$$
\begin{aligned}
R &= 4(0.1R_O + 0.2R_A + 0.5R_R + 0.2R_D) \\
P &= 4(0.5P_C + 0.2P_L + 0.1P_R + 0.2P_E) \\
C &= 0.3R + 0.7P
\end{aligned}
$$

A

Projectos computacionais

Neste Apêndice propõem-se enunciados de projectos computacionais. Estes, tal como acontece geralmente na vida real, são *abertos*, i.e., podem não especificar completamente o problema e deixar alguma margem de manobra. Por conseguinte, há quase sempre necessidade de tomar decisões e fazer opções durante o desenvolvimento do trabalho as quais devem ser sempre bem justificadas.

A.1 Aritmética computacional

A.1.1 Considere os seguintes modos de calcular o valor de $\pi = 3.141592653589793\ldots$ (ver BECKMAN (1971) ou DELAHAYE (1997) para uma perspectiva histórica):

a) Desenvolvimento de $\arctan x$ pela série de Gregory (1638-1675)

$$\begin{aligned} \arctan x &= x - \frac{1}{3}x^3 + \frac{1}{5}x^5 - \frac{1}{7}x^7 + \cdots \\ \frac{\pi}{4} = \arctan 1 &= 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \cdots \end{aligned}$$

A última expressão pode ser escrita na forma mais vantajosa (explique porquê e compare os resultados com os obtidos pela expressão original)

$$\frac{\pi}{4} = 1 - 2\left(\frac{1}{3\times 5} + \frac{1}{7\times 9} + \cdots\right)$$

b) Aplicação da fórmula de Machin (1680-1752)

$$\frac{\pi}{4} = 4\arctan\frac{1}{5} - \arctan\frac{1}{239}$$

Nota: Esta fórmula foi usada em 1949 para calcular π com 2035 dígitos no ENIAC (Electronic Numerical Integrator and Computer e foi precursora de muitas outras, conhecidas como fórmulas do tipo de Machin, algumas das quais se apresentam a seguir.

c) Aplicação da fórmula de Hutton (1737-1823)

$$\frac{\pi}{4} = \arctan\frac{1}{2} + \arctan\frac{1}{3}$$

d) Aplicação da fórmula de Clausen (1801-1885)

$$\frac{\pi}{4} = 2\arctan\frac{1}{3} + \arctan\frac{1}{7}$$

e) Aplicação da fórmula de Dase (1824-1861)

$$\frac{\pi}{4} = \arctan\frac{1}{2} + \arctan\frac{1}{5} + \arctan\frac{1}{8}$$

f) Desenvolvimento de $\arcsin x$

$$\begin{aligned} \arcsin x &= x + \frac{x^3}{2\times 3} + \frac{1\times 3\times x^5}{2\times 4\times 5} + \frac{1\times 3\times 5\times x^7}{2\times 4\times 6\times 7} + \cdots \\ \frac{\pi}{6} &= \arcsin\frac{1}{2} \end{aligned}$$

g) Método de Arquimedes (287–212 a.C.)
A área A de um círculo de raio unitário é $A = \pi$. As áreas de polígonos inscritos com $4, 8, 16, \ldots$ lados fornecem sucessivos minorantes de π. Notar que a área do triângulo inscrito cujo ângulo ao centro é θ é $1/2\sin\theta$, e que

$$\sin\theta = [(1-\cos 2\theta)/2]^{1/2}, \qquad \cos\theta = (1-\sin^2\theta)^{1/2}$$

pelo que o seno de submúltiplos do ângulo θ se pode obter indutivamente. Repita este caso com polígonos circunscritos para obter majorantes de π. (Arquimedes conseguiu chegar a um polígono com 96 lados partindo de um hexágono e o seu método manteve-se como o único utilizado até à invenção do cálculo no século XVII!)

h) Integração pela regra do trapézio
Como é fácil de ver,

$$\frac{\pi}{4} = \int_0^1 \frac{\mathrm{d}x}{1+x^2}$$

Este integral pode ser calculado aproximadamente pela regra do trapézio composta com pontos equidistantes, obtendo-se deste modo uma aproximação para π.

i) Elabore um gráfico que mostre a evolução do erro em função do número de termos empregados nos vários métodos e comente os resultados obtidos.

A.1.2 Escreva um programa para testar os métodos recursivo e compensado de calcular somatórios, tomando para x diversos valores numéricos e aplicando-os a alguns casos de teste, nomeadamente:

a) às séries

$$\exp x = 1 + x + x^2/2! + \cdots = \sum_{k=0}^{\infty} x^k/k!$$
$$\ln(1+x) = x - x^2/2 + x^3/3 - x^4/4 + \cdots$$

b) às séries do Projecto A.1.1.

Nota: Os resultados são altamente dependentes da aritmética do computador utilizado.

A.1.3 Teste os vários métodos de calcular a norma euclideana de vectores $\mathbf{x} \in \mathbb{R}^n$: começando pela aplicação directa da definição e evoluindo para os métodos sugeridos no Problema 1.9.26 de PINA (1995). Gere os vectores $\mathbf{x}$ com componentes aleatórias, não esquecendo de incluir nos testes vectores com componentes 'muito pequenas' e 'muito grandes'. Tire conclusões quanto à precisão, ao tempo de cálculo dos vários métodos e às ocorrências de *overflow* e *underflow*.

A.2 Interpolação polinomial

A.2.1 Escreva um programa para obter polinómios interpoladores na forma de Newton empregando valores nodais extraídos das tabelas seguintes.

Pretende-se utilizar este programa para obter o polinómio de menor grau que reproduza, com uma precisão especificada (erro relativo de 1%, por exemplo), os valores tabelados. Para os polinómios que satisfazem este critério, tome como indicador de qualidade o valor de $J(p)$ dado por

$$J(p) = \int_a^b \left[p''(x)\right]^2 \, \mathrm{d}x$$

em que $[a, b]$ é o intervalo de interpolação.

a)

x	10	15	20	30	40	50	60	70	80
y	3.148	2.680	2.299	1.799	1.438	1.225	1.023	0.862	0.683

b)

x	0	10	20	30	40	50	60	70	80	90	100
y	32.0	29.1	23.4	14.9	4.3	−7.4	−28.5	−36.0	−2.3	29.1	62.6

A.2.2 A pressão p e a temperatura T da Atmosfera–Padrão Internacional em função da altitude z são dadas pelos valores da tabela seguinte.

z (m)	0	500	1000	1500	2000	2500	3000	3500	4000	4500	5000
p (bar)	1.0132	.9546	.8988	.8456	.7950	.7469	.7012	.6578	.6166	.5775	.5405
T (K)	288.2	284.9	281.7	278.4	275.2	271.9	268.7	265.4	262.2	258.9	255.7

z (m)	5500	6000	6500	7000	7500	8000	8500	9000	9500	10000
p (bar)	.5054	.4722	.4408	.4111	.3830	.3565	.3315	.3080	.2858	.2650
T (K)	252.4	249.2	245.9	242.7	239.5	236.2	233.0	229.7	226.5	223.3

Escreva um programa para obter polinómios interpoladores empregando valores nodais extraídos desta tabela.

Pretende-se utilizar este programa para determinar o polinómio de menor grau que reproduza, com uma precisão especificada (erro relativo de 1%, por exemplo), os valores tabelados. Para os polinómios que satisfazem este critério, tome como indicador de qualidade o valor de $J(p)$ dado por

$$J(p) = \int_a^b \left[p'(x)\right]^2 \, \mathrm{d}x$$

em que $[a, b]$ é o intervalo de interpolação.

A.2.3 A velocidade v do som no mar depende da pressão, temperatura e salinidade em cada ponto. Esta dependência é complicada, tendo-se obtido por experimentação os seguintes valores em função da profundidade z:

z (ft)	0	500	1000	1500	2000	2500	3000	3500
v (ft/s)	5042	4995	4948	4887	4868	4863	4865	4869
z (ft)	4000	5000	6000	7000	8000	9000	10000	
v (ft/s)	4875	4875	4887	4905	4918	4933	4949	

Escreva um programa para obter polinómios interpoladores na forma de Lagrange empregando valores nodais extraídos desta tabela.

Pretende-se utilizar este programa para determinar o polinómio de menor grau que reproduza, com uma precisão especificada (erro relativo de 1%, por exemplo), os valores tabelados. Para os polinómios que satisfazem este critério, tome como indicador de qualidade o valor de $J(p)$ dado por

$$J(p) = \int_a^b \left[p'(x)\right]^2 \, \mathrm{d}x$$

em que $[a, b]$ é o intervalo de interpolação.

A.2.4 Mediram-se os seguintes valores relativos ao volume v (em litros) de líquido contido numa cisterna em função do nível z (em milímetros):

z	0	85	160	226	305	353	412	480	542
v	10	100	200	300	500	700	1000	1400	1800
z	615	700	780	870	971	1070	1181	1230	1319
v	2300	2900	3500	4200	5000	5800	6700	7100	7800
z	1371	1410	1489	1600	1690	1740	1832	1902	1960
v	8200	8500	9100	9900	10500	10800	11300	11600	11734

A cisterna tem a forma de um cilindro de eixo horizontal mas do qual se desconhece a forma geométrica da base (ver a Figura A.2.1) não podendo por isso utilizar-se um processo de integração (analítica ou numérica). Pretende-se construir uma tabela que forneça o volume v de líquido em função do nível z, de 10 em 10 milímetros, sendo essencial reproduzir os valores medidos, i.e., que constam da tabela.

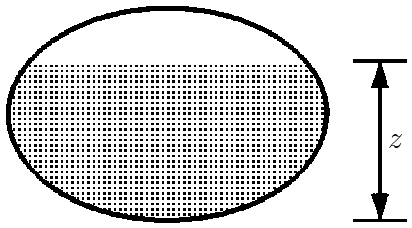


Figura A.2.1: Geometria para o Projecto A.2.4

A.2.5 Construa polinómios interpoladores da função $f(x) = 1/(1+x^2)$ no intervalo $[-5, 5]$ com nós equidistantes. Verifique que quando o grau aumenta, o polinómio interpolador apresenta oscilações cada vez mais acentuadas, e, portanto, não converge para a função interpolada dada. Repita o cálculo com os nós de interpolação de Chebyshev em vez dos nós equidistantes. Compare os resultados e tire conclusões.

A.2.6 Pretende-se construir um troço $\overline{P_0P_1}$ de ligação entre duas linha (rectas) de caminho de ferro de modo a garantir continuidade C^2 (ver a Figura A.2.2). Escreva um programa que, dadas as rectas e os pontos de ligação, determine:

a) a forma paramétrica do troço de ligação, em que $x_1(t)$ e $x_2(t)$ são polinómios apropriados:

$$\mathbf{x} = \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix}, \qquad \text{com} \quad t \in [0,1]$$

b) o raio de curvatura mínimo do troço de ligação e a respectiva localização. *Nota:* O raio de curvatura R é dado por $1/R = \|\mathbf{x}' \wedge \mathbf{x}''\|/\|\mathbf{x}'\|^3$, em que $\wedge$ denota o produto externo e $\|\cdot\|$ a norma euclideana.

A.2.7 Desenvolva um programa para o cálculo de *splines* quadráticos. Aplique este programa a alguns casos de teste, nomeadamente à interpolação das seguintes funções:

a) $f(x) = x^3 + 2x, \quad \bar{\Omega} = [0, 2]$

b) $f(x) = \sin x, \quad \bar{\Omega} = [0, \pi/2]$

c) $f(x) = (1 - x^2)^{1/2}, \quad \bar{\Omega} = [0, \sqrt{3}/2]$.

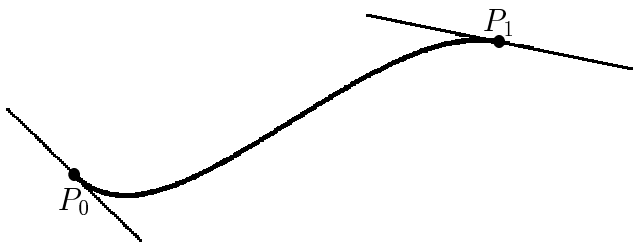


Figura A.2.2: Geometria para o Projecto A.2.6

d) Determine o erro cometido na aproximação da função f e da derivada f'. Mostre a sua evolução com o espaçamento dos nós. (Tome para y'_0 valores apropriados às funções interpoladas.)

A.2.8 Desenvolva um programa para o cálculo de *splines* cúbicos com nós equidistantes. Aplique este programa a alguns casos de teste, nomeadamente à interpolação das seguintes funções:

a) $f(x) = \cos x, \quad \bar{\Omega} = [0, \pi/2]$

b) $f(x) = (1 - x^2)^{1/2}, \quad \bar{\Omega} = [0, \sqrt{3}/2]$

c) $f(x) = x^4, \quad \bar{\Omega} = [0, 2]$

d) Determine o erro cometido na aproximação à função f e às suas derivadas f' e f''. Mostre a sua evolução com o espaçamento dos nós. (Tome para y'_0 e y'_n valores apropriados às funções interpoladas.)

e) Estime a ordem de convergência e compare-a com o valor teórico.

A.2.9 Desenvolva um programa para o cálculo de *splines* cúbicos com tensão. Aplique este programa a alguns casos de teste, nomeadamente à interpolação das seguintes funções:

a) $f(x) = \cos(2x), \quad \bar{\Omega} = [0, \pi/2]$

b) $f(x) = \ln(1 + x), \quad \bar{\Omega} = [0, 1]$

c) $f(x) = x^4, \quad \bar{\Omega} = [0, 2]$

d) Verifique a influência da tensão nos resultados obtidos.

A.2.10 Desenvolva um programa para o cálculo de cúbicas de Bézier. Aplique este programa a alguns casos de teste, nomeadamente às seguintes funções:

a) quarto de circunferência de raio 1;

b) quarto de elipse de semi-eixos 2 e 1;

c) curva, cuja equação em coordenadas polares, é $r(\theta) = 1 + \cos^2 \theta$, com $\theta \in [0, \pi/2]$.

A.2.11 Uma alternativa aos *splines* cúbicos, suficiente em muitas aplicações em Gráfica Computacional, é o de utilizar polinómios cúbicos de Hermite em cada um dos subintervalos $\Omega_i = [x_{i-1}, x_i)$, $i = 1, \ldots, n$ e estimar as derivadas $f'(x_i)$ por diferenças finitas apropriadas com vista a obter uma função de classe C^1. Desenvolva um programa para o cálculo destas funções recorrendo a diferenças finitas que utilizem os valores (x_{i-2}, y_{i-2}), (x_{i-1}, y_{i-1}), (x_i, y_i), (x_{i+1}, y_{i+1}),(x_{i+2}, y_{i+2}) e as adaptações óbvias para os pontos situados nos extremos do intervalo de interpolação. A função assim construída é, por vezes, conhecida como *subspline* de Akima e foi introduzido em AKIMA (1970). Aplique este programa a alguns casos de teste, nomeadamente às seguintes funções:

a) $f(x) = \sin x, \quad \bar{\Omega} = [0, \pi/2]$

b) $f(x) = x^4, \quad \bar{\Omega} = [0, 2]$

c) e ao seguinte conjunto de dados de AKIMA (1970):

x	0.0	2.0	3.0	5.0	6.0	8.0	9.0	11.0	12.0	14.0	15.0
y	10.0	10.0	10.0	10.0	10.0	10.0	10.5	15.0	50.0	60.0	85.0

d) Compare os resultados obtidos nas alíneas a) e b) com os valores exactos. Tire conclusões.

A.3 Derivação numérica

A.3.1 Estude o efeito do valor do parâmetro h nas fórmulas regressivas, progressivas e centrais de primeira ordem para a diferenciação numérica. Considere as funções:

a) $f(x) = \sin x^2$ para $x = 0.5$ e $x = 0.75$.

b) $f(x) = \ln(1 + \exp(\cos x))$ para $x = 2$.

c) Confirme as ordens de convergência das fórmulas e determine experimentalmente o valor óptimo de h tendo em conta os erros aritméticos. Apresente gráficos elucidativos da evolução do erro com h.

A.3.2 Escreva um programa para o cálculo da primeira derivada por diferenças centrais utilizando o processo de extrapolação de Richardson. Aplique o programa a alguns casos de teste, nomeadamente para as funções

a) $f(x) = \sin x^2$

b) $f(x) = \ln(1 + \exp(-x))$

c) $f(x) = \sqrt{1 + \sin x + x^2}$

considerando vários valores de x.

A.3.3 Para obviar os efeitos de cancelamento subtractivo, SQUIRE and TRAPP (1996) propuseram o seguinte método para o cálculo da primeira derivada de uma função $f : \mathbb{R} \to \mathbb{R}$ prolongável analiticamente. Partindo do desenvolvimento em série de Taylor (com a real),

$$f(a + \mathrm{i}\,h) = f(a) + \mathrm{i}\,h f'(a) - \frac{h^2}{2!} f''(a) - \frac{\mathrm{i}\,h^3}{3!} f'''(a) + \cdots$$

podemos tomar $f'(a) \approx \operatorname{Im} f(a + \mathrm{i}\,h)/h$ com um erro de $O(h^2)$. Escreva um programa para o cálculo da primeira derivada das funções

a) $f(x) = \sin(x^2), \quad a = 1$

b) $f(x) = x^{9/2}, \quad a = 1.5$

c) $f(x) = \exp(x)/(\sin^3(x) + \cos^3(x)), \quad a = 1.5$

por este método e por diferenças finitas centrais e compare os resultados para diferentes valores de h.

A.3.4 Escreva um programa pra obter, por diferenças finitas de segunda ordem e num ponto dado, a tangente $\mathbf{t}$, a normal $\mathbf{n}$, a binormal $\mathbf{b}$, a curvatura k e a torsão τ de uma curva em $\mathbb{R}^3$ dada pela sua representação paramétrica $\mathbf{x} = \mathbf{x}(t)$ com $t \in [0,1]$. Aplique o programa a alguns casos de teste, nomeadamente às curvas

a) $\mathbf{x} = \cos(t)\mathbf{e}_1 + \sin(t)\mathbf{e}_2$;

b) $\mathbf{x} = (1+t)\mathbf{e}_1 + t^2\mathbf{e}_2 + \exp(-t)\mathbf{e}_3$;

c) $\mathbf{x} = \sinh(1+t)\mathbf{e}_1 + \mathbf{e}_2 - t^2\mathbf{e}_3$.

em que os $\mathbf{e}_i$ denotam os versores de uma base ortogonal de $\mathbb{R}^3$.

A.3.5 Uma das aplicações mais frequentes das diferenças finitas é a de aproximar a solução de equações diferenciais. Embora este tema seja tratado no Capítulo 13 de PINA (1995), é já possível antever as possibilidades desta abordagem com um caso muito simples. Consideremos uma barra de comprimento L feita de um material cujo módulo de elasticidade é $E(x)$ com uma secção de área $A(x)$ sujeita a uma força axial $f(x)$ (ver a Figura A.3.1). Se $u(x)$ for o deslocamento axial, o equilíbrio de forças traduz-se na seguinte EDO

$$\frac{d}{dx}\left(A(x)E(x)\frac{du}{dx}(x)\right) = -f(x), \qquad 0 < x < L$$

à qual se devem juntar condições de fronteira apropriadas. No caso presente, considerar apenas deslocamentos impostos nos extremos da barra

$$u(0) = 0, \qquad u(L) = L/100$$

e $A(x) = A$ e $E(x) = E$, ambas funções constantes.

a) Adimensionalize o problema, efectuando a transformação de variáveis $x = \xi L$ e $u(x) = v(\xi)L$, obtendo uma EDO do tipo

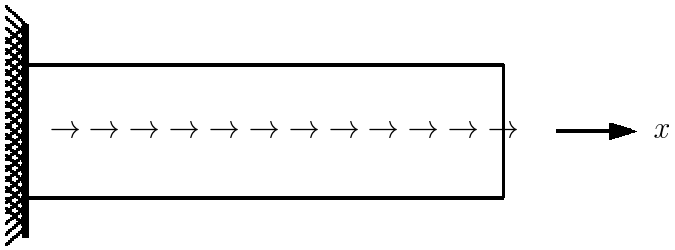
$$\frac{d^2v}{d\xi^2}(\xi) = -\phi(\xi), \qquad 0 < \xi < 1$$

b) Considere os seguintes casos de forças

$$\begin{aligned} \phi(\xi) &= \sin(\pi\xi), \qquad 0 < \xi < 1 \\ \phi(\xi) &= \begin{cases} \xi & \text{se } 0 \le \xi \le 1/2 \\ 1-\xi & \text{se } 1/2 \le \xi \le 1 \end{cases} \end{aligned}$$

c) Substitua as segundas derivadas na EDO resultante por diferenças finitas centrais numa malha uniforme, obtendo um sistema de equações algébricas com matriz tridiagonal (a ser resolvido pelo algoritmo de Thomas) e cuja solução fornecerá uma aproximação do deslocamento.

Figura A.3.1: Geometria para a EDO do Projecto A.3.5



d) Apresente gráficos mostrando a solução exacta e a solução aproximada, para vários valores do parâmetro h da malha, evidenciando a convergência.

e) *Idem*, mas agora para a derivada $dv/d\xi$ que é proporcional à tensão axial na barra.

A.4 Integração numérica

A.4.1 Escreva um programa para integração numérica de funções num intervalo $[a, b]$ dado, utilizando as regras de Gauss-Legendre compostas, considerando apenas as regras até 5 pontos. O programa deve apresentar estimativas do erro de integração baseadas na bissecção dos subintervalos. Aplique o programa a alguns casos de teste, nomeadamente:

a) $\int_{-0.5}^{3}(1+x^2)/(1+10x^6)\,\mathrm{d}x$

b) $\int_0^{\pi/2}(1-0.25\sin^2 x)^{1/2}\,\mathrm{d}x$ (integral elíptico)

c) $\int_0^1 x^{1/2}\cos(1+x)\,\mathrm{d}x$

A.4.2 Escreva um programa para integração numérica de funções num intervalo $[a, b]$ dado, utilizando *splines* quadráticos. Aplique o programa a alguns casos de teste, nomeadamente:

a) $\int_0^1 x^2\cos(1+2x)\,\mathrm{d}x$

b) $\int_0^1(\exp x)/(\cosh(2\sin x))\,\mathrm{d}x$

c) $\int_0^{\pi/2}\ln(1+x)\sin(3x)\,\mathrm{d}x$

d) Verifique a influência do valor do parâmetro m_0 nos resultados.

A.4.3 Escreva um programa para o cálculo de áreas de figuras planas convexas utilizando o seguinte método. Seja $r = r(\theta)$ a equação, em coordenadas polares, do contorno da figura. O valor da área é dado por

$$A = \frac{1}{2}\int_0^{2\pi} r^2(\theta)\,\mathrm{d}\theta$$

Deste modo, torna-se necessário calcular integrais do tipo

$$I = \int_a^b f(\theta)\,\mathrm{d}\theta \qquad \text{com} \quad f(\theta) = r^2(\theta) \quad \text{e} \quad a = 0, \quad b = 2\pi$$

A função f pode ser aproximada por um *spline* cúbico periódico para o que são dadas as coordenadas (x_i, y_i) de n pontos sobre o contorno. Aplique o programa desenvolvido a alguns casos de teste, nomeadamente ao cálculo das áreas de:

a) um círculo de raio 1;

b) elipses de semi-eixos 1 e 2^k com $k = 1, \ldots, 4$.

A.4.4 Escreva um programa para o cálculo de integrais pela regra do trapézio corrigida composta em malhas uniformes mas com as derivadas nos extremos do intervalo de integração substituídas por fórmulas de diferenças finitas que mantenham a ordem de convergência da regra. Aplique o programa a alguns casos de teste, nomeadamente:

a) $\int_0^{\pi/2} \cos x^2 \,\mathrm{d}x$

b) $\int_0^1 x(1 - \exp(x-1)) \sin x \,\mathrm{d}x$

c) $\int_{-1}^1 x^2 \ln(1+x^2)/(2-x)^2 \,\mathrm{d}x$

A.4.5 Escreva um programa para o cálculo de integrais pelo método adaptativo não-iterativo utilizando a regra do trapézio e garantindo uma precisão especificada. Aplique o programa a alguns casos de teste, nomeadamente:

a) $\int_0^{\pi/2} \sin x^2 \,\mathrm{d}x$

b) $\int_0^1 x(1.5 - x)(1 - \exp(x-1)) \,\mathrm{d}x$

c) $\int_{-1}^1 |x| \ln(2+x^2)/(2-x) \,\mathrm{d}x$

A.4.6 Escreva um programa para integração numérica adaptativa iterativa de funções num intervalo $[a, b]$ dado, empregando a regra do trapézio e garantindo uma precisão especificada. Aplique o programa a alguns casos de teste, nomeadamente:

a) $\int_0^1 \ln(2 + x^{1/2}) \,\mathrm{d}x$

b) $\int_0^1 x^{0.8}(1.2 - x)(1 - \exp|x-1|) \,\mathrm{d}x$

c) $\int_0^2 \cos[\exp(-x) + \ln(1+x^2)] \,\mathrm{d}x$

Chama-se a atenção para o facto de que o número de subintervalos em que o intervalo $[a, b]$ vai ser subdividido não é conhecido *a priori*. Nestas circunstâncias é conveniente conceber uma estrutura de dados que seja eficiente quer em memória quer em rapidez. A técnica de pilha (*stack*) satisfaz ambos estes requisitos.

A.4.7 Escreva um programa para integração numérica de funções num intervalo $[a, b]$ dado, empregando o método de Romberg. Aplique o programa a alguns casos de teste, nomeadamente:

a) $\int_0^{\pi/2} \ln(1 + 2\arctan(x/2))\ \mathrm{d}x$

b) $\int_1^2 \exp[-1/((2-x)(x-1))] \cos x\ \mathrm{d}x$

c) $\int_0^2 (1 + x^2) \sin(1 + x^3)\ \mathrm{d}x$

d) Ao cálculo do potencial P no ponto (X, Y) produzido por uma distribuição $f(x)$ de cargas eléctricas no intervalo $[a, b]$ do eixo dos xx. Compare os resultados obtidos com os valores exactos para os casos de distribuições constantes e lineares. Estude o comportamento do algoritmo quando $Y \to 0$, e $X \notin [a, b]$ e $X \in [a, b]$. O potencial é dado por

$$P(X, Y) = \int_a^b \frac{f(x)}{[(X - x)^2 + Y^2]^{1/2}}\ \mathrm{d}x$$

A.4.8 Escreva um programa para integração numérica de funções num intervalo $[a, b]$ dado, empregando a regra IMT. Aplique o programa a alguns casos de teste, nomeadamente:

a) $\int_0^1 x^{1/2} \tanh(x)\ \mathrm{d}x$

b) $\int_0^1 x^{-1/2} \cos(\pi x/2)\ \mathrm{d}x$

c) $\int_0^1 1/(1 - 0.998x^2)\ \mathrm{d}x$

d) $\int_0^2 \ln(1/x) \exp(1 + x^2)\ \mathrm{d}x$

A.4.9 Escreva um programa para integração numérica de funções altamente oscilantes num intervalo $[a, b]$ dado, empregando o método de Filon. Basicamente, este método obtém aproximações para integrais do tipo

$$I(f) = \int_a^b \cos(kx) f(x)\ \mathrm{d}x, \tag{a}$$

$$I(f) = \int_a^b \sin(kx) f(x)\ \mathrm{d}x \tag{b}$$

que surgem frequentemente em análise espectral de Fourier (ver o Capítulo 11 de PINA (1995)) considerando uma regra de integração composta numa malha uniforme com N subintervalos, sendo f é aproximada em cada subintervalo por um polinómio interpolador de grau geralmente baixo.

Considerando polinómios de grau ≤ 1 em cada subintervalo, deduza para o integral (a) que

$$I_h(f) = \frac{1}{k}\left[\sin(kb) f(b) - \sin(ka) f(a)\right] + \frac{1}{hk^2} C_n$$

$$\text{com} \quad C_n = \sum_{i=0}^{N} \gamma_i \cos(ka_i), \quad \gamma_i = \begin{cases} f(a_i) - f(a_{i+1}) & \text{se } i = 0 \\ 2f(a_i) - f(a_{i-1}) - f(a_{i+1}) & \text{se } 1 \leq i \leq N - 1 \\ f(a_i) - f(a_{i-1}) & \text{se } i = N \end{cases}$$

Deduza as fórmulas equivalentes para o integral (b).

Aplique o programa a alguns casos de teste, nomeadamente:

a) $\int_0^{2\pi} \cos(kx) x \sin x \, \mathrm{d}x$, para $k = 1, 2, 4, 8, 16$

b) $\int_0^1 \cos(100\pi x) \exp x \, \mathrm{d}x$

c) $\int_0^1 \cos(10\pi x)/(1+x^2) \, \mathrm{d}x$

A.4.10 Pretende-se determinar distâncias médias entre pontos situados numa região Ω do plano. A distância média $d_p(\mathbf{u})$ a um ponto dado $\mathbf{u}$ é definida pela relação

$$d_p(\mathbf{u}) = \frac{1}{|\Omega|} \int_\Omega \|\mathbf{u} - \mathbf{v}\|_p \, \mathrm{d}\Omega$$

e a distância média d_p entre pontos de Ω é dada por

$$D_p = \frac{1}{|\Omega|^2} \int_\Omega \int_\Omega \|\mathbf{u} - \mathbf{v}\|_p \, \mathrm{d}\Omega \, \mathrm{d}\Omega$$

em que os vectores $\mathbf{u}$ e $\mathbf{v}$ denotam as posições de dois pontos genéricos situados em Ω e $|\Omega|$ designa a área de Ω.

a) Escreva um programa para obter $d_p(\mathbf{u})$ (dado $\mathbf{u}$) para o quadrado $\Omega = [0,1] \times [0,1]$ e para as normas correspondentes aos valores $p = 1$, $p = 2$ e $p = \infty$, por integração com a regra do ponto médio composta;

b) *Idem*, para D_p;

c) Evidencie a evolução dos resultados obtidos com o número de pontos de integração e comente.

Nota: Interprete $d_p(\mathbf{u})$ como a distância média que um taxi baseado no ponto $\mathbf{u}$ percorre para atender chamadas uniformemente distribuídas numa cidade com a forma da região Ω e D_p como a média de $d_p(\mathbf{u})$ relativamente a $\mathbf{u}$. Tenha ainda em consideração que, se $I_h(f) = \sum_{i=1}^n A_i f(x_i)$ for uma regra de integração para o integral unidimensional $I(f) = \int_0^1 f(x) \, \mathrm{d}x$, então $I_h(f) = \sum_{i,j=1}^n A_{ij} f(x_i, x_j)$ é uma regra de integração para o integral bidimensional $I(f) = \int_0^1 \int_0^1 f(x,y) \, \mathrm{d}x \, \mathrm{d}y$, com $A_{ij} = A_i A_j$.

A.4.11 Um disco magnético de computador funciona basicamente do seguinte modo: a cabeça de leitura/gravação possui movimento radial e a rotação do disco providencia o deslocamento angular. Assim, usando coordenadas polares, o tempo de acesso a uma informação que está na posição $P_i = (r_i, \theta_i)$ quando a cabeça está na posição $P_c = (r_c, \theta_c)$ é dado por

$$t(P_i, P_c) = \frac{|r_i - r_c|}{c_r} + \frac{\phi(\theta_i - \theta_c)}{c_\theta}, \qquad \text{com} \quad \phi(\theta) = \begin{cases} \theta & \text{se } \theta \geq 0, \\ \theta + 2\pi & \text{caso contrário.} \end{cases}$$

em que c_r designa a velocidade radial (m/s) da cabeça e c_θ a velocidade angular (rad/s) do disco e a forma, aparentemente complicada, da função ϕ resulta do facto do disco ter um único sentido de rotação. Escreva um programa para obter:

a) O tempo de acesso médio a partir de uma posição dada da cabeça;

b) O tempo de acesso médio (em relação a todas as posições da cabeça).

Nota: Leia o enunciado do Projecto A.4.10, de que este é uma variante, e siga as respectivas indicações. Empregue valores típicos do disco do seu computador preferido admitindo, para efeitos deste projecto, que todo o disco é utilizável, i.e., não existe veio central.

A.4.12 Escreva um programa para calcular, pela regra de Simpson composta, áreas de superfícies em $\mathbb{R}^3$ dadas na forma paramétrica. Aplique o programa a alguns casos de teste, nomeadamente às superfícies

a) $\mathbf{x} = \cos(\theta)\sin(\phi)\mathbf{e}_1 + \sin(\theta)\sin(\phi)\mathbf{e}_2 + \cos(\phi)\mathbf{e}_3$, $\theta \in [0, 2\pi]$, $\phi \in [0, \pi]$ (calote hemisférica);

b) $\mathbf{x} = u\mathbf{e}_1 + v\mathbf{e}_2 + (u^3 + v^3 + u^4)\mathbf{e}_3$, $u \in [0, 1]$, $v \in [0, 1]$;

c) $\mathbf{x} = u\mathbf{e}_1 + v\mathbf{e}_2 + \sinh(1 + uv^2)\mathbf{e}_3$, $u \in [0, 1]$, $v \in [0, 1]$.

em que os $\mathbf{e}_i$ denotam os versores de uma base ortogonal de $\mathbb{R}^3$.

Nota: Leia o enunciado do Projecto A.4.10 e siga as respectivas indicações.

A.4.13 Escreva um programa para calcular, pela regra do trapézio composta, volumes de prismas em $\mathbb{R}^3$ cuja base se localiza no plano $x_3 = 0$ e a altura é determinada pela coordenada x_3 de uma superfície definida de forma paramétrica. Aplique o programa a alguns casos de teste, nomeadamente:

a) $\mathbf{x} = \cos(\theta)\sin(\phi)\mathbf{e}_1 + \sin(\theta)\sin(\phi)\mathbf{e}_2 + \cos(\phi)\mathbf{e}_3$, $\theta \in [0, 2\pi]$, $\phi \in [0, \pi]$ (calote hemisférica);

b) $\mathbf{x} = u\mathbf{e}_1 + v\mathbf{e}_2 + (u^3 + v^3 + u^4)\mathbf{e}_3$, $u \in [0, 1]$, $v \in [0, 1]$;

c) $\mathbf{x} = (u + v)\mathbf{e}_1 + v\mathbf{e}_2 + \cosh(1 + u^2 v)\mathbf{e}_3$, $u \in [0, 1]$, $v \in [0, 1]$.

em que os $\mathbf{e}_i$ denotam os versores de uma base ortogonal de $\mathbb{R}^3$.

Nota: Leia o enunciado do Projecto A.4.10 e siga as respectivas indicações.

A.5 Equações não-lineares

A.5.1 Escreva um programa para solução de equações não-lineares pelo método da falsa posição. Aplique o programa a alguns casos de teste, nomeadamente:

a) $\ln|x + \sin x| - \cos x = 0, \qquad x > 0$

b) $x^{0.8} - 2x^{0.3} - 2.5 = 0, \qquad x > 0$

c) $\tan x - x^{1/2} = 0, \qquad x \in (0, 2\pi)$

A.5.2 Escreva um programa para solução de equações não-lineares pelo método de Illinois. Aplique o programa a alguns casos de teste, nomeadamente:

a) Determinar o zero positivo do polinómio $f(x) = x^3 + 2x^2 + 10x - 20$.

b) Determinar todos os zeros reais dos polinómios

$$\begin{aligned} f(x) &= x^4 - 2.4x^3 + 1.03x^2 + 0.6x - 0.32 \\ f(x) &= 22x^4 - 12x^3 + 13x^2 - 27x - 15 \end{aligned}$$

c) Achar a menor solução positiva da equação $x = 2\sin x + 6$.

A.5.3 Escreva um programa para solução de equações não-lineares pelo método de Pegasus. Aplique o programa a alguns casos de teste, nomeadamente:

a) Determinar o zero positivo do polinómio $f(x) = x^5 + x^4 + x^3 - 8$.

b) Determinar todos os zeros reais dos polinómios

$$\begin{aligned} f(x) &= 2x^4 + x^2 + x - 6 \\ f(x) &= 30x^5 + 80x^4 + 72x^3 + 66x^2 - 29x - 91 \end{aligned}$$

c) Achar as soluções mais próximas de zero da equação $10\cos x = \exp x$.

A.5.4 Escreva um programa para solução de equações não-lineares pelo método da secante. Aplique o programa a alguns casos de teste, nomeadamente:

a) Determinar o zero positivo do polinómio $f(x) = x^6 - x^4 - x^3 - 1$.

b) Resolver a equação $\ln x = \cos x$.

c) Determinar todos os zeros reais do polinómio

$$f(x) = x^5 - 7.5x^4 + 22.5x^3 - 33.75x^2 + 25.3125x - 7.59375$$

d) Resolver a equação $x^2(x-1)^2 = 0$.

e) Produzir uma tabela de λ em função de a para $a = 0.0,\ 0.1,\ 0.2, \ldots, 2.0$ em que

$$\lambda \exp(\lambda^2) \mathrm{erf}\, \lambda = a$$

Nota: Este problema surge no estudo da fusão de um sólido semi-infinito em que λ é proporcional à velocidade da interface sólido-líquido. Para a definição da função erf e seu modo de cálculo consultar ABRAMOWITZ and STEGUN (1968).

A.5.5 Para determinação da queda de pressão em escoamentos de líquidos em tubos cilíndricos torna-se necessário obter o chamado factor de atrito f que é dado pela relação

$$\frac{1}{\sqrt{f}} = -2\ln\left(\frac{k}{3.7D} + \frac{2.51}{\mathrm{Re}\sqrt{f}}\right), \qquad \text{para} \quad \mathrm{Re} \geq 2000$$

em que D é o diâmetro do tubo, k a rugosidade e Re o número (adimensional) de Reynolds do escoamento.

Escreva uma subrotina que, dados D, k e Re, calcule f, utilizando o método da secante. Para efeitos de teste, considere os valores $k = 0.2$ mm, $D = 200$ mm e $\mathrm{Re} = 10^4, 10^5, 10^6$. *Nota:* É importante que esta subrotina seja rápida uma vez que se destina a ser chamada muitas vezes por um programa mais vasto de cálculo de instalações hidráulicas.

A.5.6 Na teoria hidrodinâmica (linear) de ondas de superfície, o comprimento de onda λ depende do período T e da profundidade h através da relação de dispersão

$$\omega^2 = gk \tanh(kh)$$

em que $\omega = 2\pi/T$ é a frequência angular, $k = 2\pi/\lambda$, o número de onda e g a aceleração da gravidade. O problema consiste em obter k uma vez conhecido ω.

a) Para obter uma forma mais conveniente para esta relação, introduza as variáveis adimensionais $\Omega = \omega^2 h/g$ e $K = kh$.

b) Mostre que ao largo (águas profundas), com $kh \to \infty$, a equação de dispersão fica $\omega^2/g = k$, o que permite determinar directamente o comprimento de onda em função do período.

c) Escreva um programa para solução deste problema pelo método de Newton, tomando como estimativa inicial a solução obtida para as condições de águas profundas. Considere os casos $T = 5, 8, 12$ s e $h = 10$ m.

A.5.7 Escreva um programa para solução de equações não-lineares pelo método de Newton. O programa deve determinar uma estimativa da multiplicidade do zero e, no caso do zero ser múltiplo, recorrer à fórmula de Schröder. Aplique o programa a alguns casos de teste, nomeadamente:

a) $x \cos x - \exp x = 0, \qquad x \in (-5, 0)$

b) $\tan x - \cos x = 0.5, \qquad x \in (0, \pi)$

c) $x = \sin x$

d) $x^m = 0, \quad m = 1, 2, 3, 4$

A.5.8 Uma variante do método de Newton consiste em aproximar a derivada da função pela derivada do polinómio interpolador da função nos pontos x_{i-2}, x_{i-1} e x_i, obtendo-se a seguinte fórmula de iteração (verificar!)

$$\begin{aligned} x_{i+1} &= x_i + h_i \\ h_i &= -f(x_i)/(f[x_i, x_{i-1}] + f[x_i, x_{i-2}] - f[x_{i-1}, x_{i-2}]) \end{aligned}$$

Escreva um programa para solução de equações não-lineares por este método. Aplique o programa a alguns casos de teste, nomeadamente:

a) $f(x) = \exp x - 2 + 0.5x^2$

b) $f(x) = \cos x - |x|^{1/2}$

c) $f(x) = 1 - x/\left[x - (\sin x)^3/6 + x^5/120\right]$

A.5.9 Escreva um programa para solução de equações não-lineares pelo método de Muller (caso real). Aplique o programa a alguns casos de teste, nomeadamente:

a) $f(x) = x^7 - 1$

b) $f(x) = x^4 - 7x^3 + 18x^2 - 20x + 8$

c) $f(x) = x^6 + 2x^5 + x^4 + 3x^3 + 5x^2 - \exp x$

d) $f(x) = x^4 - 4x^3 + 6x^2 - 4\sin x - 3$

A.5.10 Escreva um programa para solução de equações não-lineares utilizando interpolação inversa com polinómios de segundo grau. Aplique o programa a alguns casos de teste, nomeadamente:

a) $\ln(1 + x^2) - 2/(1 + x) = 0, \qquad x > 0$

b) $\cosh x = 3$

c) Determinar todos os zeros do polinómio $p(x) = x^3 + 7x^2 + 7x - 15$

A.5.11 Escreva um programa para solução de equações não-lineares pelo método de Steffensen. Aplique o programa a alguns casos de teste, nomeadamente:

a) $0.5 - x + \sin x = 0$

b) $x^3 + 2x^2 + 10x - 20 = 0$

c) $1 + \exp(-0.2x) - x^{1.5} = 0$

A.5.12 *Nota:* O objectivo deste projecto é olhar para o processo iterativo de ponto fixo $x_{k+1} = g(x_k)$ como um *processo dinâmico* em que o contador de iterações k funciona como um *tempo discreto*. De facto, iterações de ponto fixo e sistemas dinâmicos discretos são dois modos diferentes de falar de uma mesma realidade matemática.

O modelo mais simples de evolução de populações de uma dada espécie consiste em dizer que a população no instante $k + 1$ depende da população no instante k imediatamente anterior através da lei

$$x_{k+1} = rx_k \tag{a}$$

em que x_k mede, em unidades apropriadas, o tamanho da população e $r > 0$ é um paraâmetro. Nestas circunsâncias, a população evolui de acordo com a lei $x_k = r^k x_0$ em que x_0 denota a população inicial. Se $r < 1$ a população decresce até à extinção, se $r = 1$ permanece constante e se $r > 1$ aumenta exponencialmente até infinito. Ora, um crescimento ilimitado pressupõe recursos ilimitados, pelo que este modelo não é realista. Uma modificação proposta por por Verhulst em 1845 consiste em substituir (a) por

$$x_{k+1} = rx_k(1 - x_k) \tag{b}$$

que veio a ser conhecido por modelo da parábola ogística ou, simplesmente, *modelo logístico* (ver a Figura A.5.1).

a) Mostre que (a) se pode formular como um problema de ponto fixo com $g(x) = rx$ e determine, de acordo com a respectiva teoria, quais são os seus pontos fixos em função de r e a respectiva natureza (atractivos ou repulsivos).

b) *Idem*, com o mapa logístico (b).

c) Obtenha os pontos fixos de g em função de r e determine a sua natureza (atractivos, repulsivos).

d) Trace trajectórias $x_0, x_1, x_2, \ldots$, a partir de pontos iniciais x_0 diversos e confirme as previsões teóricas da alínea anterior.

e) Repita as alíneas anteriores para o mapa $x \mapsto G(x) \equiv g(g(x))$ (composição da função g consigo própria), verificando a existência de mais pontos fixos (quantos, de que natureza?) e o aparecimento de trajectórias periódicas.

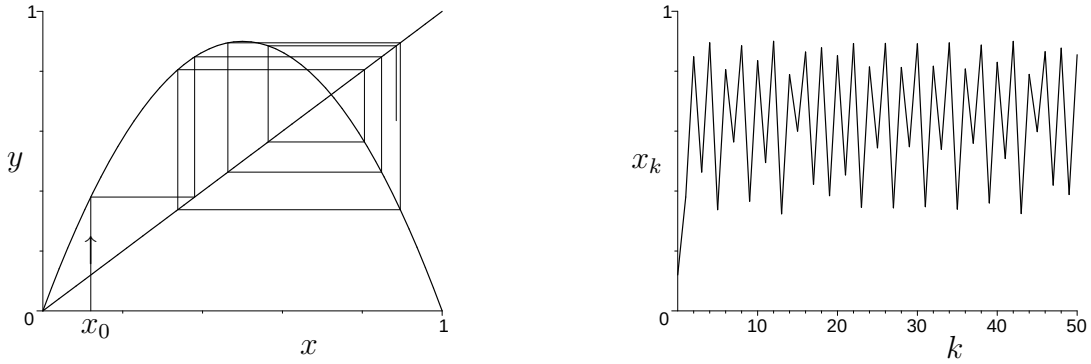


Figura A.5.1: Gráficos esquemáticos para o mapa logístico do Projecto A.5.12

A.6 Sistemas de equações lineares: métodos directos

A.6.1 Escreva um programa para a solução de sistemas de equações lineares pelo método de Gauss com pivot parcial. Aplique o programa desenvolvido a alguns casos de teste, nomeadamente sistemas com matrizes de Vandermonde (nós equidistantes) e de Hilbert de vária ordem e comente os resultados.

A.6.2 Escreva um programa para a solução de sistemas de equações lineares pelo método de Crout com pivot parcial. Aplique o programa obtido a alguns casos de teste, nomeadamente sistemas com matrizes de Vandermonde (nós equidistantes) e de Hilbert de vária ordem e comente os resultados.

A.6.3 Escreva um programa para a solução de sistemas de equações lineares $\mathbf{Ax} = \mathbf{b}$ com matrizes $\mathbf{A}$ simétricas definidas positivas recorrendo à factorização de Choleski $\mathbf{U}^T\mathbf{U}$, armazenando e utilizando apenas o triângulo superior de $\mathbf{A}$.

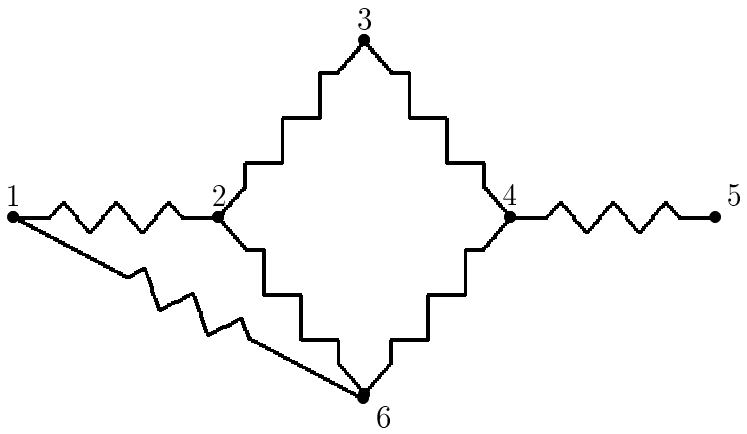


Figura A.6.1: Esquema para o Projecto A.6.5

a) Aplique o programa desenvolvido a alguns casos de teste, nomeadamente $a_{ij} = \min(i, j)$.

b) Explique o que acontece quando se aplica este método ao sistema de Wilson:

$$\mathbf{A} = \begin{pmatrix} 5 & 7 & 6 & 5 \\ 7 & 10 & 8 & 7 \\ 6 & 8 & 10 & 9 \\ 5 & 7 & 9 & 10 \end{pmatrix}, \qquad \mathbf{b} = \begin{pmatrix} 23 \\ 32 \\ 33 \\ 31 \end{pmatrix}$$

c) Teste o programa com matrizes de Hilbert de vária ordem e comente os resultados.

A.6.4 Escreva um programa para a solução de sistemas de equações lineares $\mathbf{Ax} = \mathbf{b}$ com matrizes $\mathbf{A}$ simétricas definidas positivas e em banda, recorrendo à factorização de Choleski $\mathbf{LL}^T$, armazenando e utilizando apenas a banda inferior de $\mathbf{A}$. Aplique o programa obtido a alguns casos de teste.

A.6.5 Considere uma rede eléctrica com n nós e em que cada par de nós i e j está ligado por uma resistência eléctrica R_{ij} (ver a Figura A.6.1). Pretende-se determinar as voltagens V_i em cada nó e as intensidade da corrente I_{ij} em cada resistência. A topologia da rede é dada por uma tabela que, para cada nó, lista todos os nós que a ele estão ligados. As equações resolventes estabelecem-se impondo, para cada nó, o balanço, das intensidades I_{ij} das correntes, com a convenção de sinal adequada, relacionadas com as voltagens pela lei de Ohm: $\sum_{j=1}^{n} I_{ij} = \sum_{j=1}^{n} (V_j - V_i)/R_{ij} = 0$, $i = 1, \ldots, n$, juntamente com a imposição das voltagens em dois nós.

a) Mostre que a matriz deste sistema é simétrica (fácil), tendencialmente em banda e definida positiva (recorrendo à dissipação de Joule).

b) Escreva um programa que forme e resolva o sistema de equações recorrendo ao método de Choleski para matrizes em banda.

c) Aplique o programa a redes de diferentes topologias e complexidades.

A.6.6 Escreva um programa para a solução de sistemas de equações lineares com matriz simétrica pelo método de Gauss com pivot diagonal, armazenando e utilizando apenas o triângulo superior. Aplique o programa desenvolvido a alguns casos de teste e, em particular, a sistemas com matrizes de Pascal $\mathbf{P} \in \mathbb{R}^{n\times n}$ definidas por

$$p_{1i} = p_{i1} = 1, \quad i = 1, \ldots, n; \qquad p_{ij} = p_{i,j-1} + p_{i-1,j}, \quad i, j = 2, \ldots, n$$

Nota: Estas matrizes são mal condicionadas para n elevado.

A.6.7 Escreva um programa para a solução de sistemas de equações lineares $\mathbf{Ax} = \mathbf{b}$ com matrizes $\mathbf{A}$ tridiagonais, recorrendo ao método de Thomas mas com inclusão da técnica de pivot parcial.

a) Aplique o programa desenvolvido a alguns casos de teste, nomeadamente às matrizes de Clement (HIGHAM, 1996, Apêndice E) de ordem n

$$a_{ij} = \begin{cases} i & \text{se } j = i+1 \\ n-i+1 & \text{se } j = i-1, \\ 0 & \text{nos outros casos.} \end{cases}$$

Nota: Estas matrizes são singulares para n ímpar.

b) Introduza perturbações aleatórias da ordem da unidade de arredondamento na diagonal de $\mathbf{A}$ e analise o que sucede.

A.6.8 As matrizes que a seguir se apresentam permitem ilustrar os efeitos do mau condicionamento na solução de sistemas de equações lineares $\mathbf{Ax} = \mathbf{b}$. Para tal, proceder do seguinte modo:

- Construir segundos membros $\mathbf{b}$ de modo a que $\mathbf{x} = \begin{pmatrix} 1 & 1 & \cdots & 1 \end{pmatrix}^T$ seja a solução exacta do sistema;
- Resolver o sistema pelo método de Gauss com pivot parcial, obtendo uma solução aproximada;
- Calcular os erros cometidos numa norma apropriada;
- Calcular o resíduo (não esquecer de acumular os resíduos em dupla precisão!).

a) Matriz de Hilbert.

b) $a_{ij} = (i/n)^{j-1}, \qquad i, j = 1, 2, \ldots, n.$

c) Matriz de Pei: $a_{ii} = c, \quad a_{ij} = 1 \qquad$ para $i \neq j, \quad i, j = 1, 2, \ldots, n.$
Nota: $\mathbf{A} = (c-1)\mathbf{I} + \mathbf{ee}^T$ e as dificuldades surgem para valores de c próximos de 1 ou de $1 - n$.

A.6.9 Este projecto é semelhante a A.3.5, excepto que agora se trata de encontrar a solução aproximada do deslocamento transversal de uma viga elástica encastrada numa extremidade e flectida por uma carga distribuída $q(x)$ e uma carga Q concentrada na outra

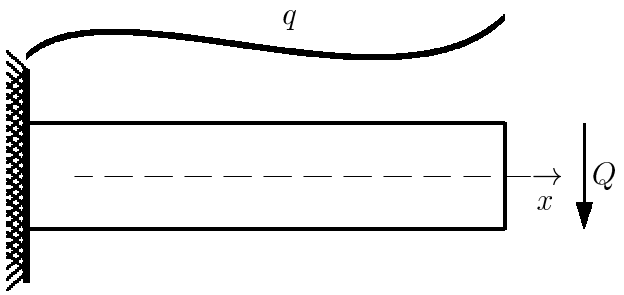


Figura A.6.2: Geometria para a EDO do Projecto A.6.9

extremidade (ver a Figura A.6.2). Se $u(x)$ for o deslocamento transversal, o equilíbrio de forças traduz-se na seguinte EDO (onde todas as variáveis foram adimensionalizadas)

$$\frac{d^4u}{dx^4}(x) = q(x), \qquad 0 < x < 1$$

No caso presente, considerar $q(x) = \sin(\pi x)$ e as seguintes condições de fronteira

$$\begin{aligned} u(0) &= 0, \qquad u'(0) = 0 \qquad \text{(encastramento)} \\ u''(1) &= 0, \qquad u'''(1) = Q = 1 \qquad \text{(carga concentrada)} \end{aligned}$$

Escreva um programa que:

a) Forme o sistema de equações algébricas lineares resultantes da substituição da quarta derivada da EDO por diferenças finitas centrais numa malha uniforme de passo $h = 1/n$ e diferenças finitas apropriadas (progressivas ou regressivas) para as condições de fronteira de modo a garantir que estas não prejudiquem a ordem de convergência. *Sugestão:* Rever o Problema 3.9.19 de PINA (1995).

b) Resolva este sistema recorrendo à factorização da respectiva matriz.

c) Compare os resultados obtidos com os valores exactos (fáceis de obter). *Nota:* O condicionamento da matriz do sistema piora com n^4.

d) Aplique a técnica de refinamento iterativo e verifique em que medida esta melhora ou não os resultados.

A.7 Valores e vectores próprios

A.7.1 Escreva um programa para a determinação do valor próprio dominante e do respectivo vector próprio de matrizes reais simétricas e em banda utilizando o método das potências directas com translação espectral, considerando o valor da translação espectral como um dado. Aplique o programa a alguns casos de teste.

A.7.2 Escreva um programa para a obtenção do valor próprio mais próximo de um dado valor e do respectivo vector próprio de matrizes reais simétricas utilizando o método das potências inversas com translação espectral. Aplique o programa a alguns casos de teste.

A.7.3 Escreva um programa para a determinação dos primeiros p valores próprios e dos respectivos vectores próprios de matrizes reais simétricas e em banda utilizando o método das iterações ortogonais. Aplique o programa a alguns casos de teste.

A.7.4 Escreva um programa para a obtenção dos valores próprios e dos respectivos vectores próprios de matrizes reais simétricas utilizando o método de Jacobi. Aplique o programa a alguns casos de teste.

A.8 Sistemas de equações sobredeterminados

A.8.1 Sabe-se que y varia com x de acordo com uma lei mais ou menos parabólica que passa pela origem. No entanto, as medições produziram os seguintes resultados:

x	0.0	1.0	2.0	3.0	4.0	5.0	6.0	7.0	8.0	9.0
y	0.0	0.5	2.0	4.5	8.1	12.3	17.8	24.8	32.5	41.3

Pretende-se fazer o ajustamento de uma função $y = u(x)$ a estes dados considerando as seguintes possibilidades:
1. $u(x) = a_0 + a_1x + a_2x^2 + a_3x^3$;
2. $u(x) = x(b_0 + b_1x + b_2x^2)$;
3. $u(x) = x^2(c_0 + c_1x)$.

a) Explicite as hipóteses que estão subjacentes a cada uma destas curvas e antecipe os respectivos méritos e deméritos.

b) Calcule os respectivos coeficientes e resíduos.

c) Avalie a hipótese de variação parabólica e tire conclusões.

A.8.2 Considere os dados do Projecto A.2.1. Determine para cada um dos casos o polinómio de menor grau que reproduz os valores dados com a precisão especificada. Utilize o algoritmo de Gram-Schmidt clássico e o modificado e compare os resultados.

A.8.3 Considere os dados do Projecto A.2.1. Determine o polinómio de menor grau que reproduz os valores dados com a precisão especificada. Recorra à factorização **QR** obtida pelo método das reflexões de Householder.

A.8.4 O calor específico c_p (em kJ/kg K) do anidrido carbónico (CO_2) a pressão constante em função da temperatura T (em ^oC) é dado pela tabela seguinte:

T	0	100	200	300	400	500	600	700
c_p	0.8148	0.9136	0.9927	1.0567	1.1103	1.1547	1.1920	1.2230
T	800	900	1000	1100	1200	1300	1400	1500
c_p	1.2493	1.2715	1.2900	1.3059	1.3197	1.3314	1.3415	1.3498

Ajuste um polinómio de grau ≤ 3 a estes valores e calcule os erros cometidos.

A.8.5 Escreva um programa para a solução de sistemas de equações lineares cuja matriz é mal condicionada recorrendo às diferentes técnicas de regularização expostas no Capítulo 8 de PINA (1995). Aplique o programa desenvolvido a alguns casos de teste de modo a evidenciar os efeitos obtidos e a sensibilidade dos resultados face aos valores dos parâmetros utilizados.

A.8.6 Sabe-se que um certo fenómeno fśico obedece a uma lei do tipo

$$y = a + b\exp(cx)$$

em que a, b e c são constantes empŕicas. Uma experiência forneceu os seguintes resultados

x	0.20	0.40	0.60	0.80	1.0	1.20	1.40	1.60	1.80
y	9.819	10.116	10.326	10.473	10.571	10.643	10.685	10.721	10.744

O carácter não linear da relação acima torna a determinação dos valores de a, b e c algo difćil. Um método possv́el para tornear esta dificuldade consiste no seguinte.
1) Substituir a relação dada por esta outra, equivalente

$$y' = A + By \qquad \text{com} \quad y' = dy/dx, \quad A = -ac \quad \text{e} \quad B = c$$

2) Obter os valores de y' nos pontos tabelados por intermédio de fórmulas de diferenciação numérica apropriadas.
3) Determinar o valor dos parâmetros A e B pelo método dos mńimos quadrados.
4) Usar um método semelhante para calcular o valor do parâmetro b.
Procedendo como se acaba de sugerir, obtenha os valores das constantes a, b e c e calcule os erros cometidos (diferença entre os valores tabelados e os calculados por este método).

A.8.7 Pretende-se ajustar uma curva do tipo

$$y = a\exp(bx)$$

ao conjunto de valores:

x	0.00	0.20	0.40	0.60	0.80	1.00	1.20	1.40	1.60	1.80	2.00
y	0.48	0.59	0.72	0.90	1.08	1.36	1.62	2.03	2.44	3.00	3.68

a) Obtenha os valores das constantes a e b por linearização e calcule os erros cometidos (diferença entre os valores tabelados e os calculados por este método).

b) *Idem*, mas resolvendo o problema não-linear.

c) Compare os resultados e comente.

A.8.8 Duas grandezas fśicas estão relacionadas por uma equação do tipo

$$y = ax^b$$

em que a e b são constantes empŕicas. Uma experiência forneceu os seguintes valores

x	1.000	1.125	1.250	1.375	1.500	1.625	1.750	1.875	2.000
y	9.80	11.71	13.76	16.02	18.08	20.75	22.93	25.89	28.77

a) Obtenha os valores das constantes a e b por linearização e calcule os erros cometidos (diferença entre os valores tabelados e os calculados por este método).

b) *Idem*, mas resolvendo o problema não-linear.

c) Compare os resultados e comente.

d) Para confirmar os valores obtidos procedeu-se a um segundo conjunto de medições, que forneceu os seguintes valores:

x	1.000	1.125	1.250	1.375	1.500	1.625	1.750	1.875	2.000
y	9.81	11.71	13.77	16.09	18.32	20.77	23.16	26.17	28.13

e) Repita as alíneas anteriores e diga quais os valores de a e b que lhe parecem mais fiáveis.

A.8.9 Num processo de arrefecimento, a temperatura u de um corpo metálico evolui com o tempo t de acordo com a lei de Newton

$$u' = \alpha u$$

em que α é uma constante propriedade do referido corpo e cujo valor se pretende determinar. Para tal efectuou-se uma experiência que forneceu os seguintes valores:

t	0.00	2.00	4.00	6.00	8.00	10.00	12.00
u	98.0	65.8	44.2	29.9	19.9	13.5	9.0

a) Calcule o valor de α e as temperaturas previstas com este valor.

b) Para testar a sensibilidade de α relativamente aos dados, suprima o valor u_i de maior discrepância e repita a alínea anterior.

A.8.10 A velocidade horizontal v do vento na *camada limite* atmosférica sobre terreno liso ou sobre o mar varia com a altitude z de acordo com a lei

$$\frac{v}{v_r} = \left(\frac{z}{z_r}\right)^a$$

em que a é um parâmetro a determinar, e v_r e z_r são valores de referência introduzidos para efeito de adimensionalização da fórmula. São conhecidos os seguintes valores obtidos por medição:

z (m)	10.	20.	30.	40.	50.	60.	70.	80.	90.	100.
v (m/s)	4.9	5.4	5.7	6.1	6.3	6.3	6.4	6.7	6.8	6.9
z (m)	200.	250.	300.	350.	400.	450.	500.	550.	600.	
v (m/s)	7.6	8.0	8.3	8.3	8.6	8.8	8.7	9.1	9.1	

a) Determine o valor de a usando o método de linearização e o método não-linear. Compare os resultados e tire conclusões.

b) Estude a sensibilidade deste valor ao valor de z_r e respectivo v_r tomados como referência.

A.8.11 Dados experimentais forneceram os seguintes valores da viscosidade μ do ar em função da temperatura:

$T\ (K)$	$\mu \times 10^5$ $(kg\,m^{-1}s^{-1})$	$T\ (K)$	$\mu \times 10^5$ $(kg\,m^{-1}s^{-1})$
200	1.360	2800	8.145
400	2.272	3000	8.516
600	2.992	3200	8.878
800	3.614	3400	9.232
1000	4.171	3600	9.579
1200	4.695	3800	9.918
1400	5.197	4000	10.252
1600	5.670	4200	10.580
1800	6.121	4400	10.902
2000	6.553	4600	11.219
2200	6.970	4800	11.531
2400	7.373	5000	11.838
2600	7.765		

Determine um polinómio que aproxime os valores dados com uma precisão, medida na norma euclidiana, pelo menos tão boa como a conseguida com a fórmula de Sutherland

$$\mu = (1.458)10^{-6}T^{1.5}/(T + 110.4)$$

frequentemente utilizada para exprimir a viscosidade de um fluido em função da temperatura.

A.8.12 Dispõem-se de medições das coordendas (x_i, y_i), $i = 1, \ldots, m$ de pontos do plano supostamente pertencentes a uma circunferência. Escreva um programa que determine o centro e o raio da circunferência que melhor se ajusta, de acordo com o método dos mínimos quadrados não-lineares, aos dados disponíveis. Aplique o programa desenvolvido a alguns casos de teste, nomeadamente:

x	5.004	4.428	2.917	1.077	−0.4271	−1.001	−0.4360	1.073	2.936	4.433
y	1.990	3.765	4.857	4.851	3.773	1.995	0.2401	−0.845	−0.856	0.2440

A.8.13 Numa fábrica são produzidas peças de forma rectangular cuja qualidade dimensional é necessário controlar. Com este objectivo, e para cada lado do rectângulo, são medidas as coordenadas (x_i, y_i) de $m \geq 2$ pontos e ajustada uma recta (aplicando-se aqui inteiramente o exposto no Problema 8.5.39 de PINA (1995)). Elabore um programa que calcule a não rectilinearidade de cada lado, medida pela distância euclidiana média à recta ajustada, o desvio dos ângulos internos relativamente a $\pi/2$ e o comprimento de cada lado, parâmetros que permitirão aceitar ou rejeitar as peças produzidas. Para efeitos de teste do programa gere as coordenadas (x_i, y_i) como perturbações aleatórias de lados de rectângulos.

A.8.14 Durante a segunda guerra mundial, os submarinos alemães detectavam os aviões aliados medindo, em intervalos regulares, o sinal de radar por estes emitidos durante a aproximação tendo em conta que a intensidade do sinal de radar decresce com o quadrado da distância. Denotando por x_i a distância na horizontal do avião ao submarino, por H a altitude e por v a velocidade do avião (ambas supostas constantes), por $\Delta t_i = t_{i+1} - t_i$ o intervalo de tempo das observações e por α_i o aumento da intensidade do sinal de radar, cada observação permite estabelecer o par de equações

$$\begin{aligned} \frac{x_i^2 + H^2}{x_{i+1}^2 + H^2} &= \alpha_i \\ \Delta t_i &= x_i - x_{i+1} \end{aligned}$$

Assim, para determinar a posição, a altitude e a velocidade do avião são necessárias pelo menos 3 observações (6 equações não-lineares). Atendendo aos erros de medição, é aconselhável efectuar mais medições que as estritamente necessárias e aplicar o método dos minímos quadrados. Escreva um programa para a solução deste problema com $N \geq 3$ observações, adoptando um processo de linearização apropriado (ver o Capítulo 10 de PINA (1995)) e aplique-o a alguns casos de teste.

A.8.15 A aproximação de dados por somas de exponenciais é um processo rodeado de perigos pois trata-se de um problema mal condicionado. Para verificar que assim é, considere os valores y_i da função $y = \exp(-x) + \exp(-2x)$ em n pontos x_i uniformemente distribuídos no intervalo $[0, 1]$ e perturbe-os aleatóriamente de modo a alterar o k-ésimo dígito. Com estes dados ajuste uma função $u = c_1 \exp(-\lambda_1 x) + c_2 \exp(-\lambda_2 x)$ pelo método dos mínimos quadrados determinando assim os parâmetros c_1, c_2, λ_1 e λ_2. Estude o que se passa tomando vários valores para n e k e tire conclusões.

A.9 Sistemas de equações lineares: métodos iterativos

Nota: Para efeitos de teste, é prático ter uma solução exacta $\mathbf{x}$ conhecida e 'simples', pelo que é usual proceder do seguinte modo: construir o segundo membro $\mathbf{b}$ do sistema $\mathbf{Ax} = \mathbf{b}$ de modo a que $\mathbf{x} = \begin{pmatrix} 1 & 1 & \cdots & 1 \end{pmatrix}^T$ seja a respectiva solução.

A.9.1 Escreva dois programas para a solução de sistemas de equações lineares tridiagonais de diagonal estritamente dominante por linhas pelos métodos de de Jacobi e de Gauss-Seidel, utilizando critérios de paragem baseados em estimativas do erro. Aplique os programas desenvolvidos a alguns casos de teste.

A.9.2 Escreva dois programas para a solução de sistemas de equações lineares pelos métodos de de Jacobi e de Gauss-Seidel, tentando acelerar a convergência por aplicação da fórmula de Aitken componente a componente de modo semelhante ao método de Steffensen. Aplique os programas desenvolvidos a alguns casos de teste e mostre o efeito da aceleração da convergência.

A.9.3 a) Determine por meio de experimentação numérica o valor óptimo do factor de relaxação para a matriz tridiagonal:

$$\mathbf{A} = \begin{pmatrix} 2 & -1 & 0 & 0 & \dots & 0 \\ -1 & 2 & -1 & 0 & \dots & 0 \\ 0 & -1 & 2 & -1 & \dots & 0 \\ & & \vdots & & & \\ 0 & 0 & 0 & \dots & -1 & 2 \end{pmatrix}$$

b) *Idem*, para a matriz pentadiagonal

$$\mathbf{B} = \begin{pmatrix} 2 & -1 & 0.5 & 0 & \dots & 0 & 0 \\ -1 & 2 & -1 & 0.5 & \dots & 0 & 0 \\ 0.5 & -1 & 2 & -1 & 0.5 & \dots & 0 \\ & & \vdots & & & & \\ 0 & 0 & 0 & \dots & 0.5 & -1 & 2 \end{pmatrix}$$

Aplique o método de Gauss-Seidel com relaxação a estes sistemas tomando como valor de arranque o vector nulo. Faça variar o factor de relaxação de 0.1 a 2 por intervalos de 0.1 e determine o erro (medido numa norma apropriada) ao fim de um número determinado de iterações (100, por exemplo). Considere o valor óptimo do factor de relaxação aquele que produziu o menor erro.

Repita estas experiências para sistemas de ordens 10, 20, 40 e 80.

A.9.4 a) Escreva um programa para a solução de sistemas de equações lineares cuja matriz é simétrica definida positiva pelo método de descida mais inclinada. Aplique os programas desenvolvidos a alguns casos de teste de modo a evidenciar a maior ou a menor rapidez de convergência do método.

b) *Idem*, para matrizes simétricas definidas positivas e em banda.

A.9.5 Escreva um programa para a solução de sistemas de equações lineares pelo método de ortogonalização de Arnoldi reiniciado após k iterações com k dado pelo utilizador. Aplique os programas desenvolvidos a alguns casos de teste de modo a evidenciar a maior ou a menor rapidez de convergência do método e avalie a influência de k na rapidez da convergência.

A.9.6 a) Escreva um programa para a solução de sistemas de equações lineares com matrizes é simétricas definidas positivas pelo método dos gradientes conjugados. Aplique o programa a alguns casos de teste de modo a evidenciar a maior ou menor rapidez da convergência do método, mostrando a evolução das normas euclidianas do erro e do resíduo ao longo das iterações.

b) *Idem*, para matrizes simétricas definidas positivas e em banda.

A.9.7 a) Escreva um programa para a solução de sistemas de equações lineares com matrizes simétricas pelo método dos resíduos conjugados. Aplique o programa a alguns casos de teste de modo a evidenciar a maior ou menor rapidez da convergência do método, mostrando a evolução das normas euclidianas do erro e do resíduo ao longo das iterações.

b) *Idem*, para matrizes simétricas e em banda.

A.9.8 Escreva um programa para a solução de sistemas de equações lineares cuja matriz é simétrica definida positiva pelo método dos gradientes conjugados com precondicionamento de Jacobi. Aplique os programas desenvolvidos a alguns casos de teste de modo a evidenciar a maior ou menor rapidez da convergência do método com e sem precondicionamento, mostrando a evolução das normas euclidianas do erro e do resíduo ao longo das iterações. Estude o efeito do número de iterações do método auxiliar.

A.10 Sistemas de equações não-lineares

A.10.1 Escreva um programa para a solução de sistemas de equações não-lineares pelo método de Newton em que a matriz jacobiana é aproximada por diferenças finitas. Aplique o programa desenvolvido a alguns casos de teste, nomeadamente:

a)

$$\begin{aligned} x^2 - xy - 20y^2 &= 1 \\ x^2 - 5xz + 6z^2 &= 0 \\ x^2y + xyz + z^3 &= 4 \end{aligned}$$

b)

$$\begin{aligned} x(\ln(y^2+1) + x) - 2 &= 0 \\ \frac{x+y+1}{y^3+2} + 1 &= 0 \end{aligned}$$

c)

$$\begin{aligned} \frac{(x+y-3)^2(y+1)}{(y-1)^2+1} - 1 &= 0 \\ (x-2)(y+1)^2 - 1 &= 0 \end{aligned}$$

A.10.2 Escreva um programa para obtenção de zeros de polinómios reais pelo método de Bairstow. Aplique o programa desenvolvido a alguns casos de teste.

A.10.3 Escreva um programa para a solução de sistemas de equações não-lineares pelo método de Newton em que a matriz jacobiana é actualizada de acordo com o método de Broyden. Aplique o programa desenvolvido a alguns casos de teste, nomeadamente aos indicados no Projecto A.5.1.

A.10.4 Desenvolva um programa para a solução de sistemas de equações não-lineares pelo método de Jacobi generalizado. Aplique o programa desenvolvido a alguns casos de teste, nomeadamente aos indicados no Projecto A.5.1.

A.10.5 Desenvolva um programa para a solução de sistemas de equações não-lineares pelo método de Gauss-Seidel generalizado. Aplique o programa desenvolvido a alguns casos de teste, nomeadamente aos indicados no Projecto A.5.1.

A.10.6 Escreva um programa para a solução de sistemas de equações não-lineares pelo método de continuação com iterações de Newton. Aplique o programa desenvolvido a alguns casos de teste mostrando a influência da escolha dos valores do parâmetro de continuação e do número de iterações de Newton intermédias.

A.11 Aproximação de funções

A.11.1 Aplique o método τ de Lanczos à solução do seguinte problema

$$-u'' + u = x^2, \qquad 0 < x < 1, \qquad u(0) = 1, \quad u'(1) = 0$$

Obtenha soluções aproximadas até ao grau 6 e compare-as com a solução exacta.

A.11.2 Desenvolva um programa para obter polinómios ortogonais num intervalo $[0, 1]$ com $p(0) = p(1) = 0$. Aplique este programa à solução aproximada de

$$-u'' + u = x^2, \qquad 0 < x < 1, \qquad u(0) = 0, \quad u(1) = 0$$

A.11.3 Desenvolva um programa que implemente as técnicas de alisamento estudadas e faça a sua aplicação a alguns casos de teste.

A.11.4 Escreva um programa para o cálculo de aproximações de Padé. Aplique-o a alguns casos de teste, nomeadamente:

a) $f(x) = \exp(x)$, em torno de $x = 0$

b) $f(x) = \sin(x)$, em torno de $x = 0$

c) $f(x) = \cos(x)$, em torno de $x = 0$

A.12 Equações diferenciais ordinárias: problemas de valor inicial

A.12.1 Escreva um programa para a solução de EDO's de primeira ordem empregando o método de Runge-Kutta de segunda ordem com controlo adaptativo de passo. Aplique o programa desenvolvido aos seguintes casos:

a) $u' = \ln|tu|, \qquad u(1) = 1, \quad t \in (1, 4]$

b) $u' = \sin(t+u), \qquad u(0) = 0, \quad t \in (0,3]$

c) $u' = \exp(-u)(t^2 + \sin u), \qquad u(0) = 1, \quad t \in (0,2]$

A.12.2 Escreva um programa para a solução de EDO's de primeira ordem empregando o método de Runge-Kutta de quarta ordem com controlo adaptativo de passo. Aplique o programa obtido aos seguintes casos:

a) $u' = 1/t^2 - u/t - u^2, \qquad u(1) = -1, \quad t \in (1,2]$

b) $u' + u = \exp(-t), \qquad u(0) = 0, \quad t \in (0,2]$

c) $u' = t - 1/u, \qquad u(0) = 1, \quad t \in (0,1]$

A.12.3 Escreva um programa para a solução de EDO's de primeira ordem empregando o método de Adams-Bashforth de terceira ordem. Aplique o programa desenvolvido aos seguintes casos:

a) $u' = 1/x^2 - u/t - u^2, \qquad u(1) = -1, \quad t \in (1,2]$

b) $u' = t + u, \qquad u(0) = 0, \quad t \in (0,1]$

c) $u' = 1 - u/t, \qquad u(2) = 2, \quad t \in (2,3]$

A.12.4 Escreva um programa para a solução de EDO's de primeira ordem empregando o método preditor-corrector de Adams-Moulton de terceira ordem. Aplique o programa obtido aos seguintes casos:

a) $u' = t - u^2, \qquad u(0) = 1, \quad t \in (0,1]$

b) $u' = 1/(t+u), \qquad u(0) = 1, \quad t \in (0,2]$

c) $u' = t - 1/u, \qquad u(0) = 1, \quad t \in (0,1]$

A.12.5 Escreva um programa para a solução de EDO's de segunda ordem, transformando estas equações num sistema de duas EDO's de primeira ordem e resolvendo este sistema pelo método de Runge-Kutta de quarta ordem. Aplique o programa desenvolvido aos seguintes casos:

a) $u'' = 2(\exp(2t) - u^2), \qquad u(0) = 0, \quad u'(0) = 1, \quad t \in (0,1]$;

b) $u'' = 2u^3, \qquad u(1) = 1, \quad u'(1) = -1, \quad t \in (1,2]$;

c) $u'' + 3u^2 = -(u')^2, \qquad u(0) = 5, \quad u'(0) = 0, \quad t \in (0,4]$;

d) $\theta'' + a\theta' + b\theta = 0, \qquad \theta(0) = \pi/4, \quad \theta'(0) = 0, \quad t \in (0,100]$.

Esta é a equação do movimento de um pêndulo com atrito em que $a \geq 0$ está relacionado com o coeficiente de atrito, e b, com a aceleração da gravidade. Usar $a = 0.1$ e $b = 10$.

A.12.6 Uma forma de dispor dos resíduos radioactivos produzidos pelas centrais nucleares adoptada por alguns paśes consiste em encerrá-los em contentores que são lançados ao mar. Há alguns anos atrás ocorreu nos EUA uma polémica sobre a segurança de tal procedimento que envolveu a AEC (Atomic Energy Commission). As dúvidas levantadas eram as seguintes:

- A hermeticidade dos contentores. A AEC conseguiu provar, mediante testes exaustivos, que os contentores não abririam fendas devido à acção da água do mar.
- A resistência ao impacte. Foi questionada a resistência dos contentores ao impacte contra o fundo do mar. Os testes efectuados mostraram que se os contentores atingissem o fundo do mar a uma velocidade superior a 12 m/s abririam fendas. A AEC negou que esta velocidade pudesse ser atingida.

O problema consiste em integrar as equações do movimento e provar ou não a correcção da afirmação da AEC. Conhecem-se os seguintes dados:

- Massa do contentor = 240 kg
- Volume do contentor = 210 l
- Profundidade do mar no local de lançamento = 100 m
- Densidade da água do mar = 1.025
- Resistência hidrodinâmica = cv com $c = 1.17$ Ns/m, e v, a velocidade de descida.

A.12.7 Pretende-se determinar as características de voo (posição, velocidade, aceleração, em função do tempo, duração e alcance) de um foguete, sendo conhecidos os seguintes dados:

- Massa inicial do foguete, incluindo combustível = 2 000 kg
- Impulso desenvolvido (constante durante 60 s) = 300 000 N
- Consumo de combustível (constante durante 60 s) = 20 kg/s
- Resistência aerodinâmica = kv^2 com $k = 0.4$ Nm^2/s^2, sendo v o módulo da velocidade
- Ângulo de disparo = 50, 60, 70 e 80 graus

A.12.8 Pretende-se determinar as características de voo (posição, velocidade, aceleração, em função do tempo) de um veículo espacial do tipo *space shuttle* admitindo os seguintes dados e condições:

- Constante de gravitação universal $G = 6.6710^{-11}$ m^3/kg s^2
- Massa da Terra $M = 5.95\ 10^{24}$kg; Raio da Terra $R = 6\,370$ km
- Coeficiente de resistência aerodinâmica $C_x = 0.83$
- Coeficiente de sustentação aerodinâmica $C_z = 0.65$
- Área de referência $S = 48$ m^2
- Massa do veículo $m = 8\,000$ kg
- Altitude inicial $z_0 = 100$ km
- Velocidade inicial $v_0 = 0.98\times$ velocidade de satelitização

- Massa específica da atmosfera em função da altitude z:

$$\rho = \rho_0 \exp(-z/z_c) \quad \text{com} \quad \rho_0 = 1.39 \text{ kg/m}^3 \quad \text{e} \quad z_c = 7160 \text{ m}$$

As forças aerodinâmicas sobre o veículo calculam-se do seguinte modo:

- Força de resistência actuando segundo a velocidade mas em sentido contrário $F_x = \frac{1}{2}\rho v^2 C_x S$.
- Força de sustentação actuando segundo a normal à velocidade $F_z = \frac{1}{2}\rho v^2 C_z S$.

Considere, para simplificar, que o veículo estava inicialmente em órbita circular equatorial e que a trajectória se desenvolve nesse plano.

A.12.9 Considere um ecossistema simples formado por coelhos e raposas. Os coelhos dispõem de alimentação em quantidade infinita, e as raposas comem os coelhos (que conseguirem apanhar!). Um modelo matemático clássico para estudar a interacção de duas espécies (predador-presa), conhecido por modelo de Lotka-Volterra, consiste no seguinte sistema de duas EDO's de primeira ordem não-lineares:

$$\begin{aligned} x' &= ax - cxy, & x(0) &= x_0 \\ y' &= -by + dxy, & y(0) &= y_0 \end{aligned}$$

em que x é o número de coelhos, y, o número de raposas; $'$ designa a derivada em ordem ao tempo t, e a, b, c e d são constantes positivas representando o comportamento das duas espécies. Se $c = d = 0$ as duas espécies não interagem, o que, nas circunstâncias deste problema, significa que os coelhos levam uma vida regalada e as raposas morrem à fome. Quando c e d forem maiores que zero, as raposas têm oportunidade de rectificar esta situação altamente injusta (do seu ponto de vista!).

a) Determine os valores de equílibro das populações, i.e, os que tornam os segundos membros das EDO's acima nulos.

b) Obtenha a solução analítica para o caso $c = d = 0$.

c) Investigue o comportamento deste ecossistema quando $a = 2$, $b = 1$ e $c = d = 0.01$ para vários valores iniciais das populações de coelhos e raposas, desde 2 ou 3 unidades até às centenas. Verifique a existência de soluções periódicas, nomeadamente quando $x_0 = 300$, $y_0 = 150$, em que o período é aproximadamente igual a 5 unidades de tempo. *Sugestão:* A representação gráfica no plano (x, y) é muito esclarecedora.

d) Avalie o efeito das seguintes políticas: permitir a caça aos coelhos, i.e., diminuir a; permitir a caça às raposas, i.e., aumentar b; permitir a caça a ambas as espécies.

e) Investigue também se há possibilidades de extinção de alguma das espécies ou, mesmo, de ambas. Considere que, se a população de uma espécie descer abaixo de um certo valor crítico, essa espécie extingue-se.

Nota: Um bom ponto de partida para o estudo dos modelos de dinâmica de populações é HABERMAN (1998).

A.12.10 Uma fábrica produz efluentes que contêm um poluente cuja concentração excede os limites normais. Para reduzir o nível de poluição, a fábrica decidiu adoptar o seguinte processo de tratamento. Os efluentes são primeiro enviados para um tanque contendo bactérias cuja finalidade é digerir o poluente e só depois são lançados num rio. Para estudar este processo, desenvolveu-se um modelo matemático baseado nas seguintes hipóteses:

- O tanque está equipado com um misturador que assegura que a concentração de poluente no tanque e à saída seja a mesma, e analogamente para a concentração de bactérias;
- As bactérias reproduzem-se proporcionalmente ao alimento de que dispõem, i.e., à concentração de poluente;
- A digestão do poluente é também proporcional à concentração de bactérias.

Sejam:

- V o volume do tanque (20 000 m^3),
- Q o caudal de efluentes (10 m^3/h),
- $p^*(t)$ a concentração de poluente à entrada do tanque (0.01 a 0.1 g/m^3),
- $p(t)$ *idem*, no tanque e à sua saída (g/m^3),
- $b(t)$ a concentração de bactérias no tanque e à sua saída (g/m^3),
- p_l limite normal estabelecido para a concentração de poluente (5×10^{-4} g/m^3),
- R a taxa de reprodução das bactérias (1.26 m^3/gh),
- M a taxa de mortalidade das bactérias ($10^{-5}/\mathrm{h}$),
- D a taxa de digestão do poluente (0.1 m^3/gh).

a) Mostre que as equações correspondentes ao modelo adoptado são

$$\begin{aligned} V\frac{dp}{dt} &= Qp^* - Qp - DVbp \\ V\frac{db}{dt} &= RVpb - MVb - Qb \end{aligned}$$

b) Confirme que o tanque possui um volume suficiente para garantir o objectivo pretendido em regime estacionário, i.e., quando $dp/dt = 0$ e $db/dt = 0$;

c) A fábrica não opera todavia sempre nas mesmas condições pelo que o caudal de efluentes e a sua concentração em poluentes podem sofrer variações em relação aos valores em regime estacionário. Pretende-se fazer uma análise de duas situações possíveis determinando em particular se o limite de poluição é excedido ou não.

- $Q = 15$ m^3/h e $p^* = 0.05$ g/m^3 durante 2 horas;
- $Q = 10$ m^3/h e um salto instantâneo do nível de poluição de 0.01 para 0.1 g/m^3.

Sempre que necessário usar como condições iniciais de p e b os respectivos valores em regime estacionário.

As autoridades sanitárias estão dispostas a tolerar que a concentração de poluente exceda p_l, mas não $2p_l$, durante um período limitado de 12 horas.

A.12.11 Duas partículas movem-se no plano exercendo uma sobre a outra forças do tipo Lennard-Jones, i.e., a força que a partícula j exerce sobre a partícula i é dada por

$$\mathbf{f}_{ij} = -F\frac{\mathbf{r}_{ij}}{r_{ij}}, \qquad F = -\frac{a_p}{r^p} + \frac{a_q}{r^q}$$

em que $\mathbf{r}_{ij} = \mathbf{x}_i - \mathbf{x}_j$ e $r_{ij} = \|\mathbf{r}_{ij}\|_2$. As partículas possuem massas $m_1 = 10$ e $m_2 = 1$, e as condições iniciais são: a partícula 1 está na origem com velocidade nula, e a partícula 2 está na posição $(10, 0)$. Determine as trajectórias de ambas as partículas (posição e velocidade) considerando várias velocidades iniciais para a partícula 2. Tome $p = 3$, $q = 5$, $a_p = 30$ e $a_q = 10$. Tenha em atenção as situações de colisão ou quase colisão (porquê?).

A.12.12 Pretende-se estudar manobras de evasão que permitam a um avião escapar a um míssil atacante. As hipóteses de trabalho são as seguintes. O avião voa à altitude H com velocidade V_a quando o míssil é disparado de uma distância D medida na horizontal. O míssil aponta permanentemente na direcção do avião e desloca-se a uma velocidade V_m constante. O tempo de voo do míssil antes de se autodestruir é T_m. A manobrabilidade do avião permite-lhe efectuar voltas com um raio mínimo de R_a e a do mísil com um raio mínimo de R_m. Considere as seguintes manobras de evasão e estude a sua eficácia:

a) Fuga com volta para trás (reacção de pânico);

b) Fuga para cima na vertical (tentativa de 'gastar' o míssil);

c) *Idem*, para baixo;

d) Apontar o avião para o míssil e desviá-lo no último instante (só para profissionais!).

Os movimentos, quer do avião quer do míssil, são executados num plano vertical. Faça as demais hipóteses que achar plausíveis para definir completamente o problema.

A.12.13 Pretende-se estudar a deformação de uma árvore de grande altura submetida ao efeito de ventos fortes e à acção do próprio peso (ver a Figura A.12.1). O comprimento do tronco é L, o peso da copa é W_L e a força do vento sobre ela D_L (despreza-se a acção do vento sobre o tronco, geralmente muito menor que sobre a copa) e estão ambos supostos concentrados na extremidade do tronco. Por seu lado, w é o peso por unidade de comprimento do tronco suposto uniforme. Numa secção transversal arbitrária, o esforço tangencial é dado por T e o esforço transverso (dirigido para a concavidade da

curva) por V. Considerando que a árvore se pode modelar como uma viga, as equações do equilíbrio de forças são

$$\begin{aligned} \frac{\mathrm{d}T}{\mathrm{d}s} - V\frac{\mathrm{d}\theta}{\mathrm{d}s} &= -w\cos\theta \\ \frac{\mathrm{d}V}{\mathrm{d}s} + T\frac{\mathrm{d}\theta}{\mathrm{d}s} &= w\sin\theta \end{aligned}$$

e a de momento flector em torno da origem

$$\frac{\mathrm{d}M}{\mathrm{d}s} = -V$$

em que s é a variável independente e representa o comprimento de arco medido a partir do topo e θ é o ângulo da tangente à curva com a vertical. De acordo com a teoria de Bernoulli-Euler para as vigas elásticas, o momento flector é dado por

$$M = -EI\frac{\mathrm{d}\theta}{\mathrm{d}s}$$

em que E é o módulo de Young e I o momento de inércia da secção da viga.

Além disso, as coordenadas da deformada são dadas por

$$\begin{aligned} \frac{\mathrm{d}x}{\mathrm{d}s} &= \sin\theta \\ \frac{\mathrm{d}z}{\mathrm{d}s} &= \cos\theta \end{aligned}$$

As EDO's acima devem ser complementadas com as condições

$$\begin{aligned} T(0) &= D_L\sin\theta(0) + W_L\cos\theta(0) \\ V(0) &= -D_L\cos\theta(0) + W_L\sin\theta(0) \\ M(L) &= 0 \\ \theta(L) &= 0 \end{aligned}$$

Estamos perante um sistema de seis EDO's de primeira ordem com condições em ambos os extremos do intervalo $[0, L]$, o que impede de considerar este problema como um de valor inicial. No entanto, para poder utilizar os métodos desenvolvidos no Capítulo 12 de PINA (1995) pode adoptar-se a seguinte abordagem que constitui aquilo que se designa por *método de pontaria*: estimar $\theta(0) \approx \theta(0)^{(0)}$ e resolver numericamente o problema de valor inicial resultante por um método apropriado (Runge-Kutta, por exemplo); se, com este valor inicial $\theta(0)^{(0)}$ se verificar que $\theta(L)^{(0)} = 0$, então agradeçamos aos deuses; se não, tentemos um novo valor, $\theta(0) \approx \theta(0)^{(1)}$ e repitamos os cálculos, e assim sucessivamente. Para valor $\theta(0) \approx \theta(0)^{(k+1)}$ pode usar-se o que se obtém por interpolação linear de $\theta(L)$ recorrendo aos valores imediatamente anteriores, $(\theta(0)^{(k)}, \theta(L)^{(k)})$ e $\theta(0)^{(k-1)}, \theta(L)^{(k-1)})$.

Nota: Este problema é adaptado de WINTER (1993, 1996) que contém informação sobre os valores numéricos das constantes necessárias e outros aspectos interessantes. Não é difícil ver que o modelo desenvolvido pode ser aplicado a torres, mastros, chaminés, etc.

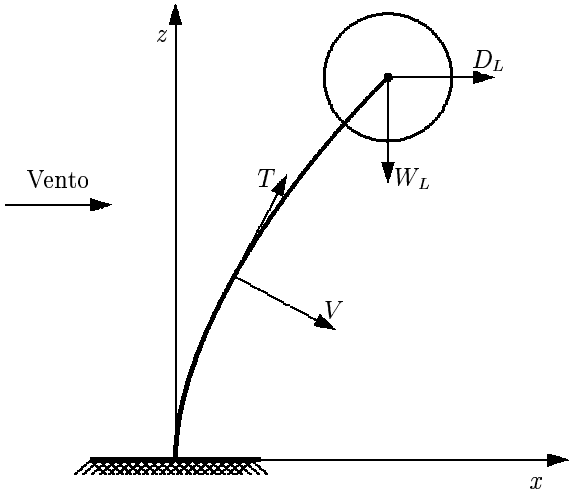


Figura A.12.1: Geometria para o Projecto A.12.13

A.13 Equações diferenciais ordinárias: problemas de valor de fronteira

A.13.1 Considere o seguinte problema

$$-\epsilon u'' + u = 1, \quad x \in \Omega = (0,1), \qquad u(0) = 0, \quad u(1) = 1$$

em que o coeficiente ϵ satisfaz $0 < \epsilon \ll 1$.

a) Obtenha a respectiva solução exacta.

b) Mostre que a maior parte da variação de u ocorre numa vizinhança da origem de espessura $O(\sqrt{\epsilon})$. *Nota:* Esta região é conhecida por *camada limite* e tem uma enorme importância em certos fenómenos físicos, como seja o escoamento de fluidos em que ϵ é proporcional à viscosidade.

c) Resolva este problema pelo método de colocação verificando o efeito do número e localização dos nós não só na solução u mas também na tensão de corte τ dada por $\tau = \epsilon u'$.

A.13.2 Repita o problema anterior mas agora recorrendo ao método dos resíduos ponderados.

A.13.3 Considere a EDO de segunda ordem com coeficientes variáveis e condições de fronteira gerais. Escreva um programa para resolver este problema pelo MEF com elementos lineares. Aplique o programa desenvolvido a alguns casos de teste de modo a evidenciar a maior ou menor rapidez da convergência do método.

A.13.4 Considere a EDO de segunda ordem com coeficientes variáveis e condições de fronteira gerais. Escreva um programa para resolver este problema pelo MDF recorrendo às fórmulas do Problema 13.9.37 de PINA (1995). Aplique o programa desenvolvido a alguns casos de teste de modo a evidenciar a maior ou menor rapidez da convergência do método.

A.13.5 Escreva um programa para mostrar que o MEF (com elementos lineares) também está sujeito a oscilações espúrias tal como o MDF. Proponha um efeito de *upwinding* por modificação das funções de forma no sentido de dar 'mais peso' a montante.

A.13.6 Em muitas situações práticas, é tão importante obter boas aproximações para u' como para u, o que geralmente não acontece se tomarmos $u' \approx u'_h$. Uma forma de conseguir este objectivo consiste em transformar a EDO de segunda ordem (13.1.1) no sistema de duas equações de primeira ordem

$$\begin{aligned} a_2 u' - p &= 0 \\ -p' + a_1 u' + a_0 u &= f \end{aligned}$$

nas variáveis independentes u e p. As aproximações baseadas nesta composição dão origem aos chamados *métodos mistos*. Aplique o método dos mínimos quadrados à minimização do resíduo deste sistema, i.e., determine as aproximações u_h e p_h em susbespaços apropriados que minimizem o resíduo

$$\|r\|^2 = \int_\Omega \left[(a_2 u' - p)^2 + (-p' + a_1 u' + a_0 u - f)^2 \right] \, \mathrm{d}x$$

Considere apenas condições de fronteira de Dirichlet e aproximações por elementos finitos lineares (ver JIANG (1998) para mais detalhes).

Aplique o programa desenvolvido a alguns casos de teste de modo a evidenciar a maior ou menor rapidez da convergência do método em ambas as variáveis u e u'.

A great discovery solves a great problem but there is a grain of discovery in the solution of every problem. Your problem may be modest; but it chalenges your curiosity and brings into play your inventive faculties, and if you solve it by your own means, you may experience the tension and enjoy the triumph of discovery.

– G. Polya (1887-1985)

How to solve it

Notas

[1]Este tema encontra-se desenvolvido em JENNINGS (1977), GOLUB and LOAN (1989) e BARRET et al. (1994).

[2]As programatecas BLAS e LAPACK constituem, no contexto da Álgebra Linear Numérica, os modelos a seguir, ver ANDERSON et al. (1995)

[3]O desenvolvimento de grandes projectos de *software* constitui hoje uma disciplina e uma indústria: a engenharia de *software*. Uma obra que reflecte o estado da arte é, por exemplo, van VLIET (2000).

[4]Uma obra interessante, escrita por quem sentiu profissionalmente o fenómeno do deslizamento do prazo dos grandes projectos de *software*, é o clássico BROOKS (1995). A sua leitura revela a dificuldade em pretender encapsular um processo tão complexo como é o desenvolvimento de *software* numa simples fórmula.

[5]As máximas que incluímos neste capítulo, como meio de sublinhar conceitos, gozam de grande popularidade entre programadores. Os seus autores são desconhecidos mas, com elevada probabilidade, foram programadores olheirentos por muitas noitadas perdidas à cata do tal último erro. De qualquer modo, SÉROUL (2000) tem o mérito de dar a estas máximas um certo ar de merecida dignidade.

[6]Para maior desenvolvimento e informação mais detalhada relativa a relatórios técnicos ou científicos recomendamos ALLEY (1996), BARRASS (1995), ou a "bíblia" TURABIAN (1996). Se o texto for de índole matemática, então HIGHAM (1998) é de consulta imprescindível.

[7]Um excelente volume dedicado ao tratamento sob forma gráfica de informação quantitativa é TUFTE (1992).

Bibliografia

M. ABRAMOWITZ and I. A. STEGUN. *Handbook of Mathematical Functions with Formulas, Graphs and Mathematical Tables.* Dover, 1968.

H. AKIMA. A new method of interpolation and smooth curve fitting based on local procedures. *J. ACM*, 17:589–602, 1970.

M. ALLEY. *The Craft of Scientific Writing.* Springer-Verlag, 1996.

E. ANDERSON et al. *LAPACK Users' Guide.* SIAM, 1995.

S. F. ASHBY, T. M. MANTEUFFEL, and P. E. SAYLOR. An extended set of basic linear algebra subprograms. *SIAM J. Numer. Anal.*, 27(6):1542–1568, 1990.

R. BARRASS. *Students Must Write: A Guide to Better Writing in Coursework and Examinations.* Routledge, 1995.

R. BARRET et al. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods.* SIAM, 1994.

P. BECKMAN. *A History of π (PI).* St. Martin's Press, 1971.

F. P. BROOKS. *The Mythical Man-Month.* Addison-Wesley, 1995.

J.-P. DELAHAYE. *Le Fascinant Nombre π.* Pour la Science, Diffusion Belin, 1997.

J. J. DONGARRA, J. DuCROZ, S. HAMMARLING, and R. HANSON. An extended set of basic linear algebra subprograms. *ACM Transactions on Mathematical Software*, 14(1), 1988.

G. H. GOLUB and C. F. VAN LOAN. *Matrix Computations.* The Johns Hopkins University Press, 1989.

R. HABERMAN. *Mathematical Models: Mechanical Vibrations, Population Dynamics and Trafic Flow.* SIAM, 1998.

N. J. HIGHAM. *Accuracy and Stability of Numerical Algorithms.* SIAM, 1996.

N. J. HIGHAM. *Handbook of Writng for the Mathematical Sciences*. SIAM, 1998.

A. JENNINGS. *Matrix Computations for Engineers and Scientists*. J. Wiley, 1977.

B. JIANG. *The Least-Squares Finite Element Method*. Springer-Verlag, 1998.

D. A. PATTERSON and J. L. HENNESSY. *Computer Architecture – A Quantitative Approach*. Morgan Kaufmann Publishers, 1996.

H. PINA. *Métodos Numéricos*. McGraw-Hill de Portugal, 1995.

R. SÉROUL. *Programming for Mathematicians*. Springer-Verlag, 2000.

W. SQUIRE and G. TRAPP. Using complex variables to estimate derivatives of real functions. *SIAM Review*, 40(1):110–112, 1996.

V. STRASSEN. Gaussian elimination is not optimal. *Numerische Mathematik*, 14(3):354–356, 1969.

E. R. TUFTE. *The Visual Display of Quantitative Information*. Graphics Press, 1992.

K. L. TURABIAN. *A Manual for Writers of Term Papers, Theses, and Dissertations*. The University of Chicago Press, 1996.

H. van VLIET. *Software Engineering: Principles and Practice*. J. Wiley, 2000.

D. F. WINTER. On the stem curve of a tall palm in a strong wind. *SIAM Review*, 35(4):567–579, 1993.

D. F. WINTER. Erratum and reformulation: On the stem curve of a tall palm in a strong wind. *SIAM Review*, 38(3):485–486, 1996.

Índice